FMC

February 1990

Report No. R-6376

ONR Final Report

A Blackboard-based Dynamic Instructional Planner

William R. Murray

Artificial Intelligence Center
Corporate Technology Center
FMC Corporation
1205 Coleman Avenue, Box 580
Santa Clara, California 95052

Sponsored by

Manpower, Personnel and Training R&D Committee of the Office of the Chief of Naval Research

Air Force Human Resources Laboratory

Naval Training Systems Center

Naval Personnel Research and Development Center

Under contract N00014-86-C-0487

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release, distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): R-6376	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: FMC, Corporate Technology Ctr Artificial Intelligence; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: Office of Naval Research - Cognitive Science Program
6c. ADDRESS (City, State, and ZIP Code): FMC Corporation, CTC 1205 Coleman Avenue Santa Clara, CA 95050	7b. ADDRESS (City, State, and ZIP Code): Office of Naval Research 800 North Quincy Street (Code 1142PT) Arlington, VA 22217-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable): 222	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: N00014-86-C-0487
8c. ADDRESS (City, State, and ZIP Code): 800 North Quincy Street Arlington, VA 22217-5000	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. 62233N; PROJECT NO. RM33M20; TASK NO.; WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification): A Blackboard-based Dynamic Instructional Planner

12. PERSONAL AUTHOR(S): William R. Murray

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 9/15/86 TO 10/31/89	1990, February 20	70

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Control for Intelligent Tutoring Systems, Instructional Planning, Dynamic Planning, Blackboard Architecture
05	08		
12	05		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This research explores dynamic planning as a control mechanism for intelligent tutoring systems. The motivation for this research is a desire to integrate plan-based and opportunistic approaches to instruction to provide more effective and versatile tutoring systems. Although planned instruction is not always required, planning can provide more coherent instruction, more effective time management, and greater responsiveness to student needs than unplanned instruction. Furthermore, a dynamic planner allows pedagogical knowledge to be applied *during* instruction rather than requiring a curriculum author to anticipate student performance and then pre-store appropriate tutorial responses. The more economic representation of pedagogical knowledge in the planner facilitates extension to new domains compared to a CAI system, which procedurally

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL: Dr. Susan Chipman	22b. TELEPHONE (Include Area Code): (202) 696-4318; 22c. OFFICE SYMBOL: ONR 1142CS

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

encodes pedagogical decisions. The planner is also better suited to handling the combinatorics of tutorial situations in which mixed-initiative instruction is allowed, a fine-grained student model varies, time is limited, and there are many alternative instructional actions.

The Blackboard Instructional Planner is a blackboard-based dynamic planner for intelligent tutoring systems. Although experimental, the planner demonstrates key plan generation and replanning capabilities required to handle common tutorial situations. It generates a sequence of lesson plans customized to a student model inferred from a pre-instruction questionnaire. The content, delivery, and length of each lesson is determined by the inferred student model, by the time allotted for lessons, by the target skill to be taught, and by the subject domain. These lesson plans are revised during instruction in response to student questions and requests, changes in time remaining for lessons, and modifications to the student model.

This report focuses on the *final* version of the Blackboard Instructional Planner. Section 7, Related Work, explains differences between this version and the earlier version, and between this research and earlier project research on generic ITS architectures. A complete guide to all project publications is included in Section 8.3, Research Contributions, on page 57.

The planner is designed to be generic to tutors that teach troubleshooting for complex physical devices. It has been implemented as the controller for the Lower Hoist Tutor, a prototype tutor for the Mark-45 naval gun mount, to demonstrate the planner's operation and means of integration. The tutor teaches troubleshooting of the lower hoist assembly, a complex hydraulic-electronic-mechanical assembly of the Mark-45, by first imparting a mental model of the device and its operation. A STEAMER-based display of the lower hoist schematic has been adapted for use in exposition, assessment, and troubleshooting practice. Lesson plan steps are instructional procedures that can use this interface to interleave the presentation of explanatory text with the highlighting and animation of icons that represent device parts. Other assessment procedures monitor student progress by asking questions or observing student task performance. Insufficient student progress initiates replanning.

This research contributes to an understanding of dynamic instructional planners, planner-controlled tutors, and ITS control architectures. The planner implementation shows precisely how a blackboard architecture can be used to realize a dynamic instructional planner. The plan representation and planning approach provide for plan customization to student background and three kinds of adaptive replanning (for requests, time, and the student model). The tutor implementation demonstrates how such a planner can be embedded in an intelligent tutoring system and what the respective roles of the different components of a planner-controlled tutor are. Finally, the analysis of the planner's use of the blackboard architecture clarifies requirements for control architectures in intelligent tutoring systems and trade-offs made in choosing alternatives.

Abstract

This research explores dynamic planning as a control mechanism for intelligent tutoring systems. The motivation for this research is a desire to integrate plan-based and opportunistic approaches to instruction to provide more effective and versatile tutoring systems. Although planned instruction is not always required, planning can provide more coherent instruction, more effective time management, and greater responsiveness to student needs than unplanned instruction. Furthermore, a dynamic planner allows pedagogical knowledge to be applied *during* instruction rather than requiring a curriculum author to anticipate student performance and then pre-store appropriate tutorial responses. The more economic representation of pedagogical knowledge in the planner facilitates extension to new domains compared to a CAI system, which procedurally encodes pedagogical decisions. The planner is also better suited to handling the combinatorics of tutorial situations in which mixed-initiative instruction is allowed, a fine-grained student model varies, time is limited, and there are many alternative instructional actions.

The Blackboard Instructional Planner is a blackboard-based dynamic planner for intelligent tutoring systems. Although experimental, the planner demonstrates key plan generation and replanning capabilities required to handle common tutorial situations. It generates a sequence of lesson plans customized to a student model inferred from a pre-instruction questionnaire. The content, delivery, and length of each lesson is determined by the inferred student model, by the time allotted for lessons, by the target skill to be taught, and by the subject domain. These lesson plans are revised during instruction in response to student questions and requests, changes in time remaining for lessons, and modifications to the student model.

The planner is designed to be generic to tutors that teach troubleshooting for complex physical devices. It has been implemented as the controller for the Lower Hoist Tutor, a prototype tutor for the Mark-45 naval gun mount, to demonstrate the planner's operation and means of integration. The tutor teaches troubleshooting of the lower hoist assembly, a complex hydraulic-electronic-mechanical assembly of the Mark-45, with the goal imparting a mental model of the device and its operation. A STEAMER-based display of the lower hoist schematic has been adapted for use in exposition, assessment, and troubleshooting practice. Lesson plan steps are instructional procedures that can use this interface to interleave the presentation of explanatory text with the highlighting and animation of icons that represent device parts. Other assessment procedures monitor student progress by asking questions or observing student task performance. Insufficient student progress initiates replanning.

This research contributes to an understanding of dynamic instructional planners, planner-controlled tutors, and ITS control architectures. The planner implementation shows precisely how a blackboard architecture can be used to realize a dynamic instructional planner. The plan representation and planning approach provide for plan customization to student background and three kinds of adaptive replanning (for requests, time, and the student model). The tutor implementation demonstrates how such a planner can be embedded in an intelligent tutoring system and what the respective roles of the different components of a planner-controlled tutor are. Finally, the analysis of the planner's use of the blackboard architecture clarifies requirements for control architectures in intelligent tutoring systems and trade-offs made in choosing alternatives.

Acknowledgements

This project was sponsored jointly by the Manpower, Personnel and Training R&D Committee of the Office of the Chief of Naval Research (under contract N00014-86-C-0487); the Air Force Human Resources Laboratory; the Naval Training Systems Center; and the Naval Personnel Research and Development Center. I would like to thank our contract monitors, Kurt Steuck (AFHRL) and Susan Chipman (ONR), for their technical advice and guidance.

This project and the ideas presented here have evolved from my interaction with colleagues over the past three years. Perry Thorndyke first proposed using a blackboard architecture for dynamic instructional planning. Stuart Macmillan and Derek Sleeman later proposed a blackboard architecture for dynamic instructional planning with the ability to improve its own planning behavior. Their work, focussing on architectural issues, is presented in [MacMillan 87] and [MacMillan 88]. Subsequent work by the author focussed on implementing a planner for a naval training domain within the existing BB1 blackboard architecture. The first version of the planner is described in [Murray 89a] and this report describes the second and final version. I would like to thank N.S. Sridharan, Perry Thorndyke, and Barbara Hayes-Roth for their advice on this project; Lee Brownston for assisting in the implementation; and both Agustin Araya and Tom Hester for reviewing drafts of this report. Thanks also to our friends in Minneapolis: Tom Hoffman, of FMC's Naval Systems Division, who served as our subject matter expert; and John Darvish and Bob Jancoski, of FMC's Advanced Systems Center, who implemented the STEAMER graphics rendition of the lower hoist schematic.

Trademarks

'Symbolics' and 'MacIvory' are trademarks of Symbolics, Inc. 'Explorer' is a trademark of Texas Instruments, Inc.

Table of Contents

List of Figures

1. Introduction - Motivation and Objectives

The goal of this research is to increase the flexibility and effectiveness of computer-based tutoring systems by applying dynamic planning techniques to their control. Both traditional CAI systems and current intelligent tutoring systems tend to be quite limited in their ability to generate customized lesson plans, execute them, and respond flexibly and appropriately to new information about the student, student questions and requests, and time remaining in lessons. These limitations and our approach to overcoming them are discussed below.

Traditional CAI systems do not plan at all. Instead they follow lesson plans authored by humans (see Figure 1-1). These may be very well-crafted plans and the instruction may be effective, but the student has little control over the interaction. For example, the student usually cannot interrupt the instruction with questions about the domain or requests to review or cover particular material. These questions and requests must be disallowed, or severely restricted, since the CAI tutor cannot reason about the domain or its lesson plan during instruction. Instead, all such questions and requests must be anticipated by the curriculum author. Most instructional customization is confined to branching within the lesson plan since there is no fine-grained student model.[1] Opportunistic teaching methods are avoided since the tutor has impoverished problem-solving capabilities and so cannot readily evaluate partial student solutions. Even if it could determine when to intervene, the lack of domain expertise and a fine-grained student model would make it difficult to determine what to say.

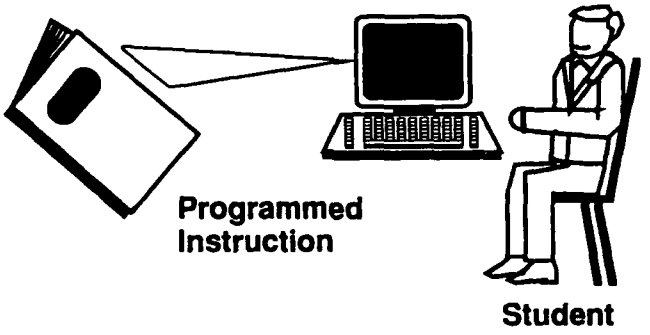


Figure 1-1: Traditional computer-assisted instruction

On the other hand, most intelligent tutoring systems (see Figure 1-2) rely

[1]Incorporating a fine-grained student model and allowing a wide range of student questions and requests significantly increases the combinatorics of tutorial situtations that the curriculum author must anticipate.

primarily on opportunistic teaching approaches. Their domain expertise and student modeling capabilities support these approaches and distinguish these systems from traditional CAI systems. Because these tutors rely on opportunistic approaches they typically assume that primary instruction has been delivered elsewhere. Generally, lesson planning is not done by these tutors although recent systems have focused on sophisticated local discourse planning. This lack of a lesson planning capability limits the ability of these tutors to generate and deliver customized expository instruction (either primary or remedial), to manage their time well, and to respond to requests to alter or explain lesson content or delivery. The purpose of the Blackboard Instructional Planner is to integrate these two complementary means of instruction - plan-based and opportunistic teaching approaches - for tutors and domains where this is most appropriate.

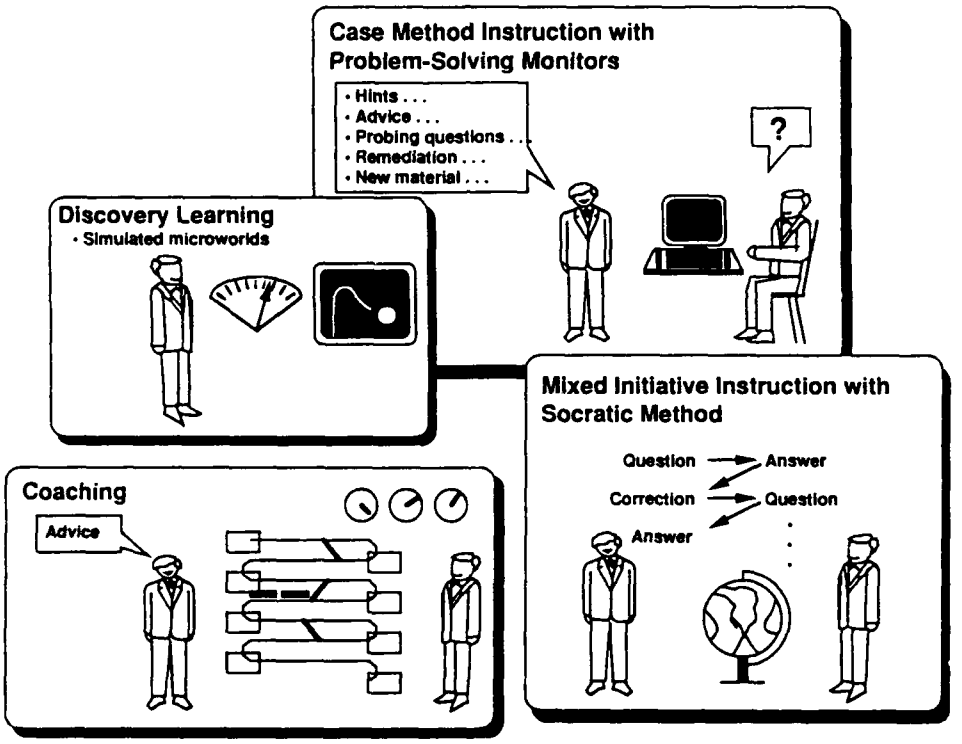


Figure 1-2: Opportunistic teaching approaches of intelligent tutoring systems

Our approach to this control problem is called *dynamic instructional planning*. In this approach, an initial instructional plan is generated by the planner. This plan is customized to an inferred student model. It also takes into account the resources available to the tutor (e.g., time). The tutor interprets this plan to control its delivery of instruction. The planner is dynamic since it can

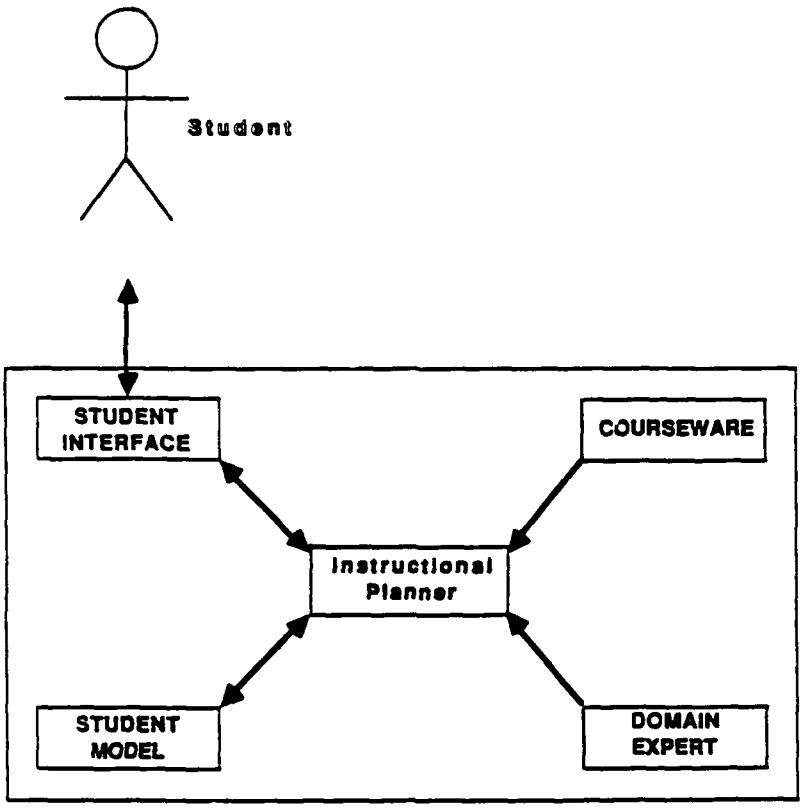


Figure 1-3: Relationship of the planner to other ITS components

later revise this plan *during* instruction as the tutorial situation changes. The tutorial situation changes as the student model changes, the amount of time available changes, and as student-initiated interactions interrupt the tutor's plan.

The planner operates as the control element of an intelligent tutoring system, as shown in Figure 1-3 (arrows indicate data flow). The planner generates an initial instructional plan customized to the inferred student model. The actions in the instructional plan are procedures that control the text, highlighting, and animation displayed on the *student interface*. The interface also accepts student input, including student-initiated questions and requests that can interrupt these *instructional procedures*. The library of possible instructional procedures the planner can draw upon is part of the *courseware*. The courseware also includes curriculum materials such as test questions and troubleshooting cases. The *domain expert* evaluates student performance on these cases, demonstrates correct troubleshooting, and provides answers to student questions about the domain. These evaluations update the *student model*. Changes to the student model cause replanning if student progress is much better or worse than expected.

The chief contribution of this research is the Blackboard Instructional

Planner, a blackboard-based dynamic instructional planner for intelligent tutoring systems. It generates a sequence of lesson plans customized to a student model inferred from a pre-instruction questionnaire. The content, delivery, and length of lessons are determined by the inferred student model, by the time allotted for lessons, by the target skill to be taught, and by the subject domain. These lesson plans are revised during instruction in response to student questions and requests, changes in time remaining for lessons, and modifications to the student model.

The planner presented in this report is actually the second version of the Blackboard Instructional Planner. It will be abbreviated as BB-IP, unless there is a possibility of confusing it with the earlier planner, in which case it will be referred to as BB-IP-2. The earlier planner will always be abbreviated as BB-IP-1.

Now we consider the research objectives of this project in more detail. They were to:

1. *Develop a dynamic instructional planner* - capable of both customized plan generation and dynamic plan revision in response to
    a. *Changes in the student model*
    b. *Changes in time remaining*
    c. *Student questions and requests*
2. *Demonstrate feasibility for a training application* - i.e., show that the planner can be embedded in a tutor, generate appropriate lesson plans, respond appropriately to student initiative, and achieve near real-time performance. All instructional actions must be fully implemented, not just simulated.
3. *Demonstrate utility of approach* - i.e., show that the planner can generate finely-tuned lesson plans, allow flexible mixed-initiative instruction, and integrate customized expository instruction with case-method problem solving and microworld exploration.

These objectives have been achieved as will be described in the remainder of this report. To circumscribe the project goals, the following were *not* research objectives:

1. *To build a complete tutor for the training application* - the focus was just on the planner. Enough of the other components were implemented to suggest how a complete tutor would operate, but much more work is required before such a tutor is ready for classroom instruction.
2. *To prescribe pedagogical theories of planning and replanning* -

> although the current implementation encodes specific means of plan customization and replanning, these can be changed. Alternate theories could be incorporated in the planner, resulting in different plan generation and replanning behavior. The *planner framework*[2] - the plan representation, resources for planning, and use of the blackboard architecture for generation, execution, monitoring, and replanning - would be unchanged.

The Blackboard Instructional Planner has been incorporated into the Lower Hoist Tutor, a prototype tutor for the Mark-45 naval gun mount, to demonstrate the planner's operation and means of integration. Before describing the planner and tutor further, a brief overview of the lower hoist domain will be helpful. The Mark-45 is a 5-inch 54-calibre gun, about six stories high, composed of over 23,000 individual parts and 15 major assemblies. It is shown in Figure 1-4. The U-shaped assembly at the bottom that connects the ammunition handling room with the loader room is called the *lower hoist*. The lower hoist assembly has a simple function: to raise or lower rounds of ammunition from the magazine to a mechanical storage drum where rounds are stored prior to firing. Its actual mechanism is complex. It consists of electrical solenoids and switches; hydraulic valves and pistons; and mechanical latches, gears, and a drive coupling. A schematic is shown in Figure 1-5. The mechanism is complex since the lower hoist can operate in several different modes of operation and a latching mechanism is required to secure rounds in position when the device is not operating.

With this background the target functionality of the planner can be illustrated with examples from the lower hoist domain. Figure 1-6 illustrates the ability to generate a customized sequence of lesson plans for a particular student, taking into account individual background and time constraints. The top and bottom parts of Figure 1-7 illustrate the tutor's presentation being interrupted with either a request or a question. In each case the tutor must decide whether to accommodate the student's question or request, to defer it, or to explain why it cannot be handled. Figure 1-8 illustrates the tutor monitoring student performance in troubleshooting. When performance is less than expected the tutor alters its instructional approach to provide remedial instruction. These three examples illustrate the most important abilities of the current planner - plan generation, mixed-initiative instruction, and adaptive replanning. Eleven

---

[2]The word "framework" is used rather than "architecture" to imply that the current planner is generic to troubleshooting tutors and extensible but that it does not possess the characteristics of an architecture. Specifically, there is no precise programming discipline and the planner is not a shell into which domain knowledge can be readily added.

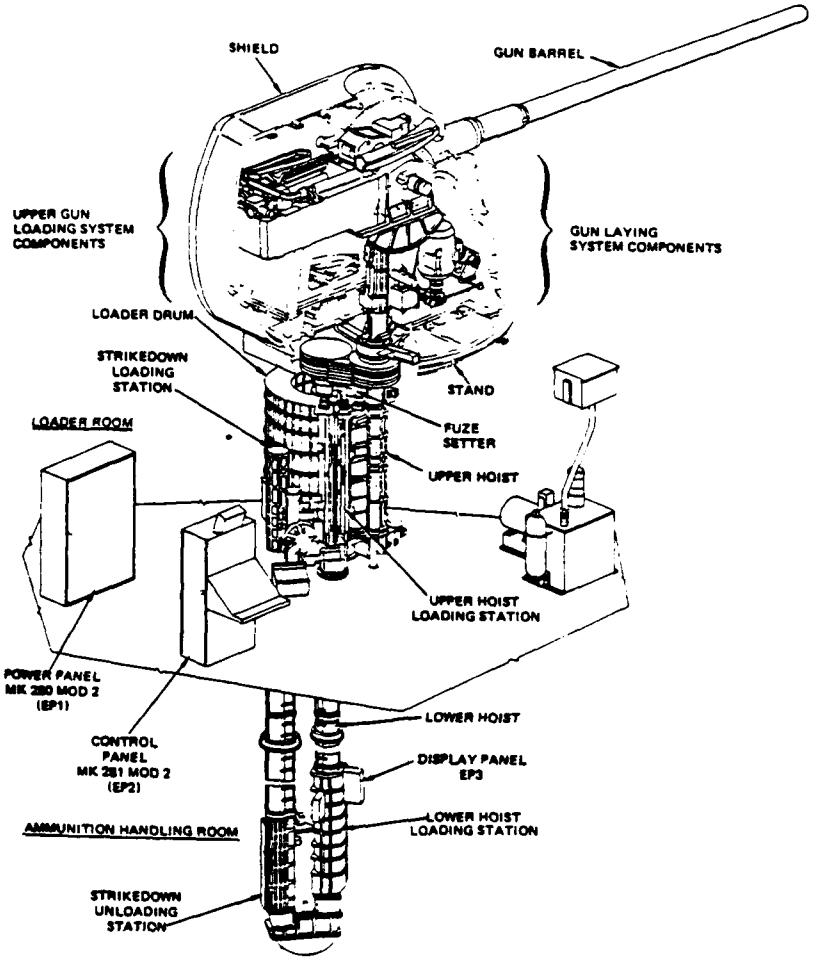


Figure 1-4: The Mark-45 naval gun mount

scenarios illustrating these capabilities and others (e.g., incremental planning and time management) have been implemented in the lower hoist domain for the Blackboard Instructional Planner and Lower Hoist Tutor.

These scenarios have been chosen to be representative of the planning and replanning situations that would arise for a planner-controlled tutor. Each scenario illustrates some aspect of planner functionality such as time management, question handling, request handling, or providing remedial instruction as needed. A scenario assumes particular student behavior - such as asking a question during an instructional procedure or performing poorly on an assigned troubleshooting task. The scenarios are not hardwired since similar

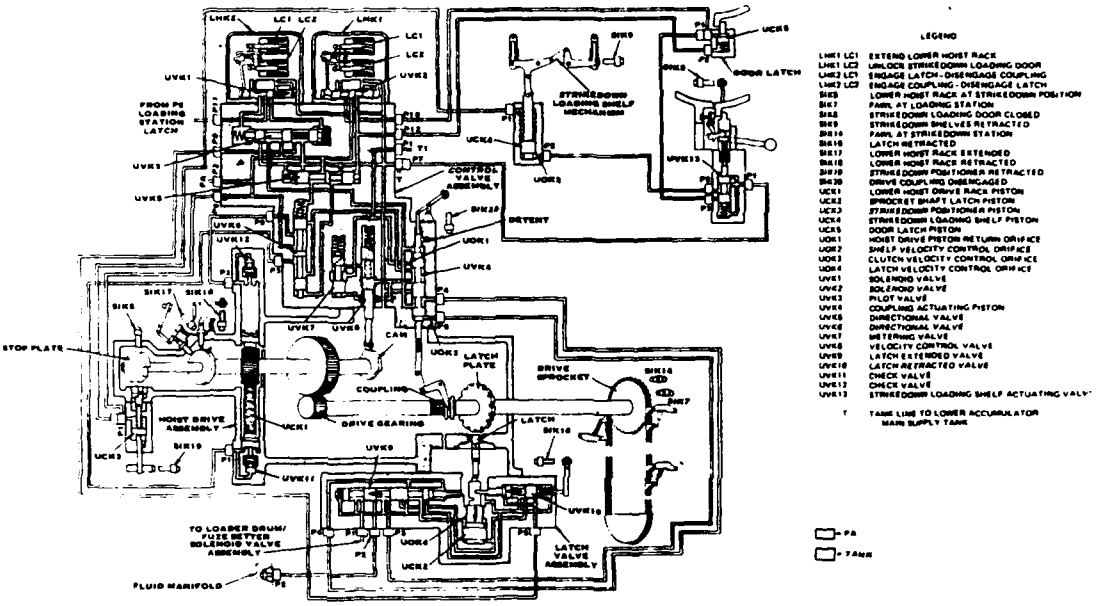


Figure 1-5: Schematic of the lower hoist assembly

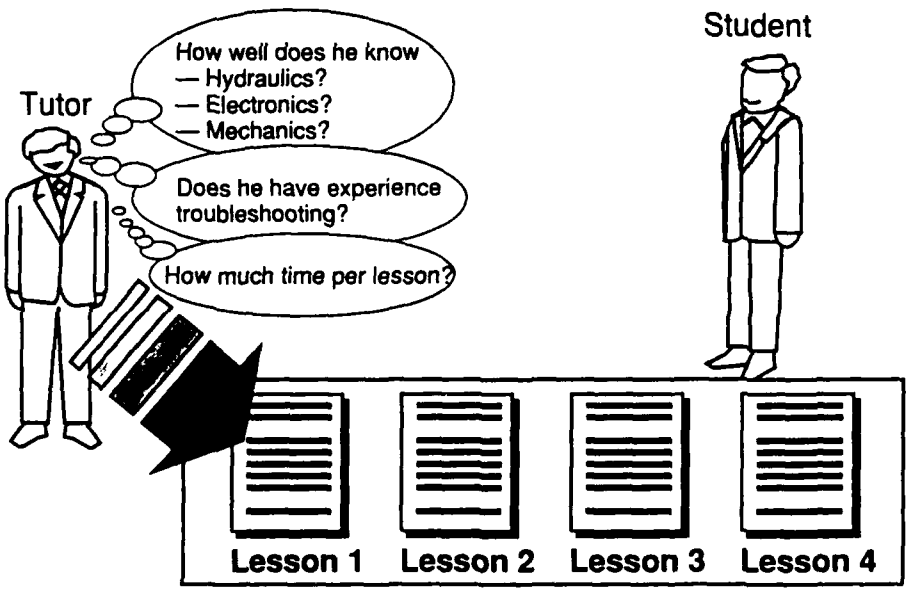


Figure 1-6: Customizing lesson plans

student behavior can occur elsewhere in the lesson plan and produce appropriate responses. For example, in the question-handling scenario the student's question

is not restricted to just one particular question that must be asked during just one particular instructional procedure. Instead, any question can be asked in any instructional procedure. The knowledge sources for handling these scenarios are specific to the planner functionality involved, not the scenarios. On the other hand, when all knowledge sources are enabled and the Lower Hoist Tutor is run as a tutor per se it frequently tends to replan when it should not, and not replan when it should. So the Lower Hoist Tutor testbed is more than just a set of hardwired scenarios, but is still quite far from a tutor ready for student use.

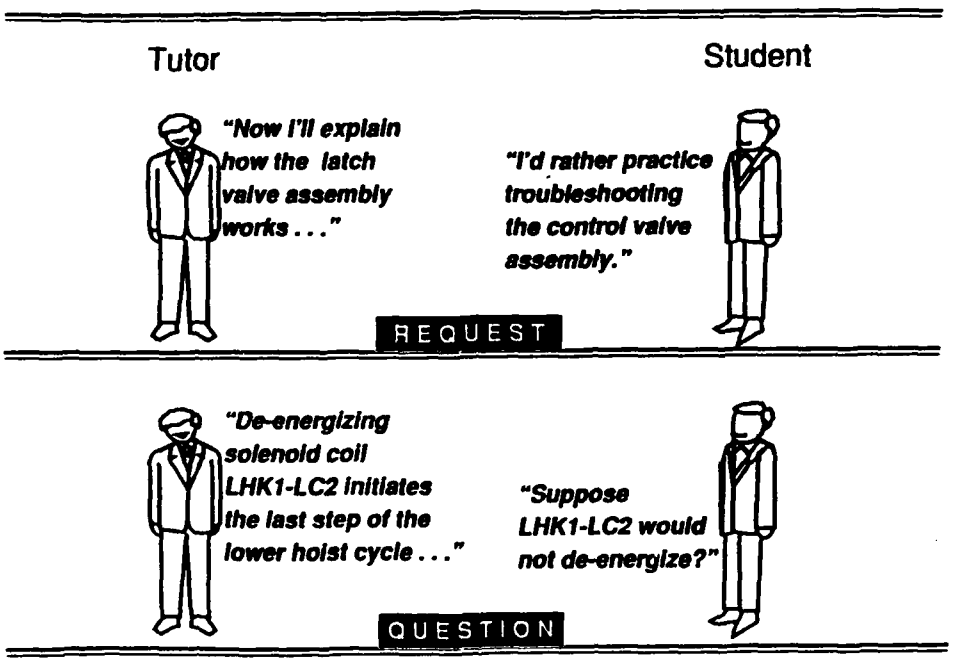


Figure 1-7: Handling questions and requests

The remainder of this report describes the implementation of the planner and the tutor, and the role of the blackboard architecture in the implementation. Section 2 provides an overview of the planner. Section 3 describes the other components of the Lower Hoist Tutor and the integration of the planner with these components. Details of plan representation, generation, execution, and replanning are discussed in Sections 4 and 5. Section 6 discusses the role of the blackboard architecture. The final two sections discuss related work, conclusions, and future directions for research.

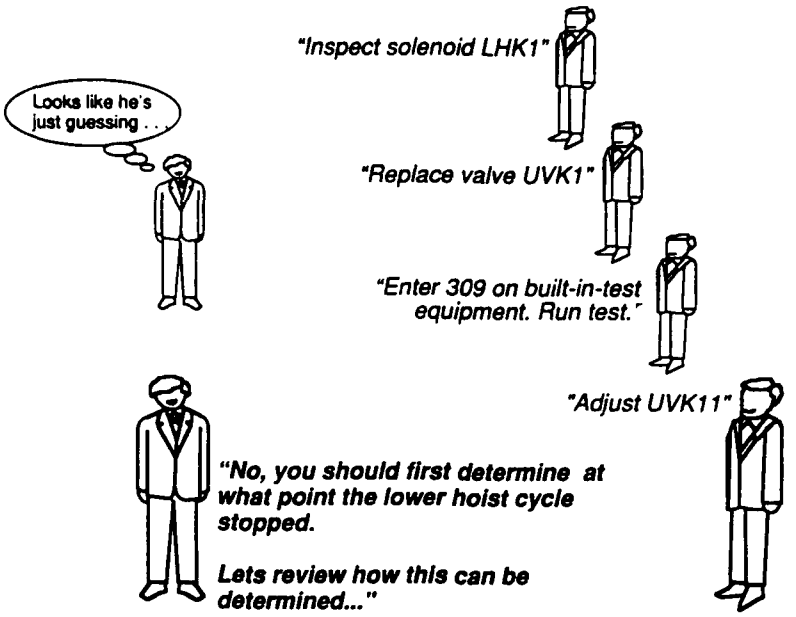


Figure 1-8: Adaptive replanning

2. Overview of the Blackboard Instructional Planner

This section provides an overview of the Blackboard Instructional Planner (BB-IP), showing what the planner does by examples but deferring details until later sections. Different scenarios are used to illustrate plan generation, question handling, time management, and replanning to provide remedial instruction. Before discussing the examples, we consider the scope of the planner to describe to what degree it is generic.

BB-IP is neither domain dependent nor domain independent, rather it was designed for a class of tutoring systems. These are tutors that teach troubleshooting for complex hydraulic-electronic-mechanical systems, where effective troubleshooting requires a mental model of device operation. A further assumption is that the tutor has a STEAMER [Hollan 84] simulation of the device that it can use for exposition and assessment. Thus, BB-IP could be used in tutors for other assemblies of the Mark-45 and similar complex defense equipment. However, BB-IP has only been tested with the lower hoist assembly and no authoring tools have been implemented. It is not clear how much transfer there would be if BB-IP were applied to tutors outside of its intended class. Much of the planner - such as plan representation, and use of blackboard architecture - would transfer. However, the set of generic troubleshooting skills that BB-IP

draws upon in its plan generation process would not apply. A skill breakdown for the new domain would be required. So it would be more difficult to apply BB-IP to programming tutors or foreign language tutors, although much of the current planner design should be reusable.

With this background we consider examples of BB-IP in operation. Two types of plans are shown in the scenarios below. A *lesson plan* is shown as a sequence of steps where each step is an *instructional procedure* to be performed. For example, such a procedure might present text explaining the operation of some part or it might give a quiz over its role. The overall sequence of lesson plans is referred to as the *instructional plan*. The process of producing such a sequence of lesson plans will be referred to as *curriculum planning*. Both kinds of plans will be shown in a considerably simplified form compared to their actual internal representation. Such plans have a richer hierarchical structure representing both abstract lesson plan steps and the rationale behind these steps, as discussed in Section 4.

2.1. Examples of Plan Generation and Customization

First, we consider plan generation and customization. In all of the examples presented we assume the following options are selected by the instructor:

- *Device knowledge base* - is the Mark-45 lower hoist assembly. This knowledge base is a semantic representation of the parts breakdown and the changes that occur in normal operation of the device.
- *Instructional objective* - is CAN-TROUBLESHOOT-LOWER-HOIST, the highest level skill. Instructional objectives can also be low level, such as CAN-IDENTIFY-PART-UVK3 or intermediate level, such as UNDERSTANDS-STRUCTURE-LOWER-HOIST.
- *Time constraints* - are 45 minutes per lesson with hard deadlines, meaning that lessons cannot exceed 45 minutes. Soft deadlines allow lessons to run over or under so that overall lesson length averages about 45 minutes. Any other lesson length can also be chosen.
- *Pedagogical mode* - is PRIMARY-INSTRUCTION. The tutor can also REVIEW a skill or let the student PRACTICE a skill.
- *Planning mode* - is COMPLETE-PLAN-ELABORATION rather than INCREMENTAL. In the latter the choice of procedures for a lesson plan step is deferred until the previous step has been executed.

All these options can be changed by menu.

There are 24 basic student models consisting of a combination of cognitive stereotypes and a general assessment of student aptitude. For each of the areas of

hydraulics, electronics, or mechanics the student can be either deficient in that area or have the prerequisite knowledge.[3] The student's aptitude is an orthogonal measure. It is either low, average, or high according to the student's overall learning rate. The cognitive stereotypes and aptitude are inferred based on a series of multiple-choice questions that precede instruction. Different sets of questions are given for hydraulics, electronics, and mechanics.

Appendix B provides two instructional plans to illustate plan customization and the steps in an instructional plan. The first plan is for a low aptitude student lacking proper skills in electronics. This student will be referred to as ELECTRONICS-DEFICIENT, LOW-APTITUDE. The second plan is for a high aptitude student with the expected background in hydraulics, electronics, and mechanics. This second student will be referred to as HAS-PREREQUISITES, HIGH-APTITUDE. Before we consider differences between the two plans we will consider the content of the first plan.

The first instructional plan consists of five lessons and 51 steps. In the first lesson an introduction to lower hoist structure and operation is given, followed by a detailed description of the lower hoist parts, their location, and their role in lower hoist operation. Part of the overview of the operation of the lower hoist is shown in the color photographs of Figure A-1. The second lesson explains how solenoids operate and then provides a detailed explanation of all the part state changes that occur in a normal cycle of operation of the lower hoist. This detailed explanation interleaves text explaining part state changes with animation showing the changes discussed. Some of this explanation is shown in Figure A-2 (color photographs (a) through (g)). In the third lesson the student must demonstrate his understanding of normal operation by successively pointing to the next part that should change state and indicating what the new state should be. Essentially, the student leads the simulation. In the fourth lesson, the tutor discusses the possible faults that can occur in the lower hoist. It also discusses how to reason from symptoms to possible faults. Then it presents a troubleshooting strategy that takes into account part change cost, malfunction probability, and split-half testing. Finally, troubleshooting is demonstrated for a particular case. The last lesson is devoted to letting the student practice troubleshooting for various cases with the tutor's assistance.

The lessons for the HAS-PREREQUISITES, HIGH-APTITUDE student are customized differently. Instead of five lessons and 51 steps there are only three lessons and 26 steps. This plan omits much of the layered instruction given to the

---

[3]These particular stereotypes were selected based on advice from our subject matter expert, taking into account his means of tailoring instruction to students.

first student. For example, the procedures that introduce the structure and operation of the lower hoist in a layered fashion (the *Cycle-overview* and *Structure-Overview* procedures referred to in Appendix B) are omitted. Similarly, procedures that introduce troubleshooting in a layered fashion (such as *Explain-Fault-Types* and *Generating-Plausible-Fault-Hypotheses*) are also omitted along with topics that the tutor judges to be of lower priority, such as part locations and descriptions. Procedure parameters also differ in the procedure that monitors student troubleshooting (*Monitor-Student-Troubleshooting*). In the second instructional plan its parameters are set to provide hints later, tolerate more incorrect actions, and provide less assistance with selecting test codes than in the first instructional plan.

2.2. Dynamic Replanning

BB-IP is a dynamic planner precisely because most instructional plans will need to be changed eventually. It might seem that if this is the case it would be better not to plan at all. But then the tutor might not finish before time (or the student's patience) was exhausted, the student could be confused as the tutor wandered from topic to topic (perhaps with some local coherence), and the importance of topics would not necessarily be reflected in the dialog or instruction given. Given enough time everything might be taught but this is an unrealistic expectation. A more detailed discussion of the advantages of having an instructional plan and being able to modify it during the instructional session is presented in [Murray 89b]. Below we consider three examples of dynamic replanning illustrating mixed-initiative instruction, time management, and the diagnosis and correction of an ineffective instructional plan.

2.2.1. An Unexpected Question

Consider an example where a question is asked that interrupts the tutor's presentation. Assume the tutor is executing *Explain-Subcycles* (step 21) of the first instructional plan. It is explaining the detailed sequence of changes that occur during the operation of lower hoist - the LHK2-LC2 solenoid coil energizes, solenoid valve UVK1 shifts right, pilot valve UVK3 shifts right, coupling actuating piston UVK4 shifts down... During this explanation the student interrupts with a question, as shown in Figures A-2 (h) through (j).

The interruption and specification of questions are handled by a menu interface as shown. Procedures are themselves broken down into smaller steps, allowing student questions and requests to occur between these *procedure steps*. The steps in the *Explain-Subcycles* procedure correspond to the presentation of text chunks explaining part state changes followed by incremental STEAMER display updates illustrating part state changes. Assessment procedures that ask a

series of questions (short answer, true/false, or multiple choice) can also be interrupted between questions. This step-wise implementation of procedures ensures that the student will never have to wait long for an opportunity to interrupt and that procedures are only interrupted at points that allow easy resumption.

At any of these interruption points the student can select to either continue, ask a question, or make a request. In this example the student selects "Ask a Question" and then a second menu appears showing templates for various questions that can be asked. These templates are drawn from the various instructional procedures that can be used to answer questions. Possible templates are retrieved from each instructional procedure in the instructional procedure library, which is part of the courseware. In this way the tutor restricts the student to only ask questions that it can answer by running instructional procedures. For example, the template associated with the procedure *Part-Roles* is "What is the role of ...?". The student selects this template from the menu of question templates and then points to coupling actuating piston UVK4. This sequence is shown in Figures A-2 (h), (i), and (j).

The tutor can either defer or directly answer the question. In this case the tutor chooses to answer the question now since the material being asked about will not be covered later. To answer the question the current procedure is suspended and a step to answer the question is inserted. The step *Explain-Subcycles* **[device subcycles lower hoist]** is suspended and two steps are spliced in after it:

Patch-1:Part-Roles(to-answer-Question #1) **[uvk4]**
Patch-1:Explain-Subcycles(continued) **[device subcycles lower hoist** *continued***]**

The first step answers the question and the second continues *Explain-Subcycles* where it was interrupted. But this is not all that happens since the student model is affected simply by the student asking the question.

When the student asks a question about material that was covered earlier, the tutor's belief that the student knows that material is diminished. In this case the tutor covered the roles of the lower hoist parts in step 10 (the *Part-Roles* procedure). The tutor detects that its prior goal, that the student knows the roles of these parts, is no longer satisfied and still should be. To reachieve this goal it splices in a review of the roles of the lower hoist parts before resuming *Explain-Subcycles*. Now the affected portion of the plan, after both plan patches have been added, appears as:

Explain-Subcycles **[device subcycles lower hoist]**
Patch-1:Part-Roles(to-answer-Question #1) **[uvk4]**

Patch-2:Part-Roles(to-provide-review) **[parts lower hoist]**
Patch-2:Multiple-Choice-Quiz **[part roles lower hoist]**
Patch-1:Explain-Subcycles(continued) **[device subcycles lower hoist** *continued*]

Patch-2:Multiple-Choice-Quiz has been added to check that the tutor's review is successful.

Now plan critics are applied to improve the flow of tutorial discourse. There is an abrupt context shift from the review of the roles of the lower hoist parts back to the detailed discussion of the lower hoist cycle. This shift in discourse is smoothed out by adding a step

Patch-2:Transition **[part roles, device subcycles lower hoist]**

whose execution is shown in Figure A-2 (n). Another plan critic attempts to remove redundant discussions. It ensures that only the first *Part-Roles* procedure discusses uvk4 by removing uvk4 from the list of parts discussed by the second *Part-Roles* procedure. After all the edits, the affected portion of the lesson plan appears as shown in Figure 2-1.

Explain-Subcycles
Patch-1:Part-Roles(to-answer-Question #1) **[uvk4]**
*Patch-2:Part-Roles(to-provide-review)***[parts lower hoist except uvk4]**
*Patch-2:Multiple-Choice-Quiz***[part roles lower hoist]**
Patch-2:Transition **[part roles, device subcycles lower hoist]**
Patch-1:Explain-Subcycles (continued) **[device subcycles lower hoist** *continued*]

Figure 2-1: Instructional plan after question is asked

2.2.2. Less Time than Expected

The question asked by the student has an additional side-effect other than altering the student model. It alters the amount of time left in the lesson. Now we consider an example where the amount of time remaining in a lesson is less than the time required to finish the lesson.

Assume the tutor is executing *Explain-Predicting-Symptoms-from-Faults*, which was originally step 34. The tutor estimates that it has about 37 minutes left in this lesson and that 27 minutes are needed to finish the activities it has planned. Suppose the amount of time remaining is reduced to only 10 minutes by manually altering the amount of time left. This can be done using the planner interface shown in Figure 2-2, which the student would not have access to or see.

The tutor detects that the amount of time remaining is insufficient to finish the remaining steps of the lesson. It removes the remaining partitions between lessons and then determines where partitions should be placed considering the

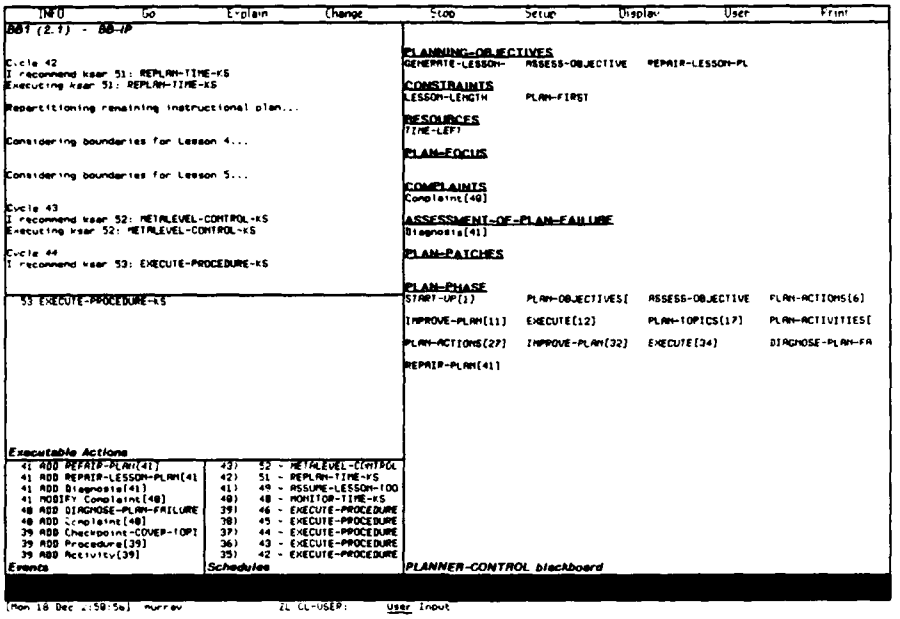


Figure 2-2: Planner interface, after replanning for time available

amount of time available now. The result is that the steps:

Short-Answer
Summarize-Topic
Demo-Troubleshooting

are moved from the end of lesson four to the beginning of lesson five. Actually, the partition between lesson four and five is moved, but the effect is the same. Figure 2-2 shows the planner interface after these operations. That part of the instructional plan that has been changed is now shown in Figure 2-3. For comparison, the original instructional plan is shown in Appendix B.

2.2.3. A Failing Instructional Objective

Failure to achieve or make adequate progress towards an instructional objective can also trigger replanning. This replanning is caused by changes to the student model. Continued poor student performance lowers the tutor's belief that the student is learning the skill currently being taught. When this happens, the planner attempts to determine why the instructional plan is failing. Then it replans to address the problem.

For example, assume the student is practicing troubleshooting in the procedure *Monitor-Student-Troubleshooting* of the last lesson. The student is expected to select troubleshooting actions to diagnose the fault in the lower hoist

34. *Explain-Predicting-Symptoms-from-Faults*
35. *Multiple-Choice-Quiz* **[propagating faults to symptoms lower hoist]**
36. *Summarize-Topic* **[predicting faulted behavior lower hoist]**
37. *Multiple-Choice-Quiz* **[predicting faulted behavior lower hoist]**
38. *Motivate-Topic* **[abduction lower hoist]**
39. *Generating-Plausible-Fault-Hypotheses* **[generating the fault hypothesis set lower hoist]**
40. *Short-Answer* **[generating the fault hypothesis set lower hoist]**
41. *Summarize-Topic* **[abduction lower hoist]**
42. *True-False-Quiz* **[abduction lower hoist]**
43. *Motivate-Topic* **[device troubleshooting lower hoist]**
44. *Explain-Troubleshooting-Strategy* **[troubleshooting strategy weighted split half troubleshooting]**
45. *Wrap-Up* **[lesson 4]**

Lesson 5

46. *Overview* **[lesson 5]**
47. *Short-Answer* **[troubleshooting strategy weighted split half troubleshooting]**
48. *Summarize-Topic* **[device troubleshooting lower hoist]**
49. *Demo-Troubleshooting* **[troubleshooting strategy weighted split half troubleshooting]**
50. *Monitor-Student-Troubleshooting* **[case-difficulty 3, wrong-tries-before-hint 2, prompt-menu yes, number-of-cases 5]**
51. *Course-Wrap-Up*

Figure 2-3: Instructional plan after replanning for available time

and then repair it. However, the student repeatedly selects poor actions and requests numerous hints. For each simulated troubleshooting action the tutor compares the student's choice to the domain expert's to compute a measure of utility. This measure of utility is used to update the student model. Over a series of poor choices, the tutor incrementally lowers its belief that the student has

acquired the skill of troubleshooting the lower hoist. When the difference between the expected change to the student model (an increase) and the actual change (a decrease) is too large then the tutor suspends execution of the instructional plan. It notes that the current instructional plan is not having its expected effect, i.e., it is failing. The tutor enters a diagnosis phase. The current approach to diagnosis is to determine the prerequisite skills for the current instructional objective, order them by their likelihood of being misunderstood, and then to assess the student's knowledge of each prerequisite. For each prerequisite, the assessment is performed by asking multiple choice questions. When the tutor finds a prerequisite that the student does not know, it infers that the problem with the current plan is that the student has forgotten or never learned that prerequisite. To correct this problem it splices in remedial instruction for the prerequisite.

In this example the skill is *Can-Troubleshoot-Lower-Hoist* and there are numerous prerequisites such as *Understands-Operation-Lower-Hoist*, *Understands-Device-Subcycles-Lower-Hoist*, etc. Some skills the tutor assumed that the student had because of the inferred student model. When the tutor asks questions about one of these skills, *Can-Predict-Part-Type-Operation-Piston*, the student cannot answer the questions correctly. The tutor infers that this is the missing prerequisite causing the student's poor performance. It splices in a discussion of the operation of pistons, a quiz, and then a transition back to the troubleshooting practice as shown in Figure 2-4.

Monitor-Student-Troubleshooting **[case-difficulty 3, ...]**
Patch-3:Explain-Part-Type-Operation[**piston**]
Patch-3:Multiple-Choice-Quiz [**operation pistons**]
Patch-3:Transition [**operation pistons, device troubleshooting lower hoist**]
Patch-3:Monitor-Student-Troubleshooting(continued)

Figure 2-4: Instructional plan after diagnosis and remediation

Of course this is just one approach to diagnosis and revision of a failing instructional plan. The fault in the plan may actually be different - the student might not be motivated or the tutor's representation of the skill being taught may differ from the student's. Alternative approaches to diagnosis that have been implemented are to simply ask the student what skills he thinks that he needs to review, or simply to reteach the failing instructional objective using a different approach. The creation of multi-step diagnostic plans to pinpoint the problem is possible, but not implemented.

The key point in the three examples presented here is that the planner and tutor functionality illustrated by Figures 1-6, 1-7, and 1-8 has been implemented. In order to demonstrate these kinds of functionality specific decisions had to be

made about how to customize plans, how to handle student questions, and how to diagnose and correct ineffective plans. The claims made in this report are not about these specific decisions, since at present there is little guidance from the educational psychology literature that is specific enough to be applied in these situations. Other approaches can be implemented with the planner that would customize plans differently and make different decisions about when and exactly how to replan. The claim made is that the planner framework - the plan representation, the use of blackboard architecture, and the kinds of plan editing that can be performed - would support alternative pedagogical approaches as well.

2.3. Other Planning Capabilities

The three scenarios above are fully implemented along with eight others demonstrating different types of planner functionality. Here is the full set of eleven scenarios used to test the planner:

Lesson Plan Generation

1. For LOW-APTITUDE, ELECTRONICS-DEFICIENT student.
2. For HIGH-APTITUDE, HAS-PREREQUISITES student.
3. REVIEW only for HIGH-APTITUDE, HAS-PREREQUISITES student.
4. PRACTICE only for HIGH-APTITUDE, HAS-PREREQUISITES student.

Dynamic Replanning

5. For a failed instructional objective.
6. In response to a student request.
7. In response to a student question.

Curriculum Planning and Time Management

8. Basic curriculum planning.
9. Not enough time.
10. Too much time.

Incremental Planning

11. Incremental planning.

The fifth, seventh, and ninth scenarios illustrating replanning were used as the examples of Section 2.2. The other scenarios are discussed below.

The two other plan generation scenarios (3 and 4) demonstrate that the tutor can be used in three ways:

1. *To teach material for the first time* - as illustrated in the instructional plans discussed in Section 2.1.
2. *To review material covered earlier* - for example, to provide a refresher course several months after initial instruction.
3. *To provide practice on some skill* - for example, the student can practice troubleshooting or predicting part state changes.

When the tutor is used to review material the only difference in its planning is that it is more selective in the topics it covers. It only reviews the most important topics, skipping over some of the topics it would have covered in primary instruction. When the tutor is used in the practice mode it acts as a problem-solving monitor, although it will still plan and deliver remedial instruction as necessary. It only puts into the initial lesson plan those procedures necessary to practice the requested skill along with any auxiliary procedures needed to improve the discourse flow.

The request handling scenario (the sixth) is similar to the question handling scenario, but the replanning can result in steps being omitted from the lesson plan, not just added. In this scenario the student requests to use the device simulation while the tutor is giving a detailed explanation of part state changes with *Explain-Subcycles*. The student's request is granted by splicing in a step (the *Explore-Device-Simulation* procedure) that lets the student use the device simulation while the tutor monitors the student's actions. Another step to continue the *Explain-Subcycles* procedure where it was interrupted is spliced in after this new step. Next, the student-requested activity *Explore-Device-Simulation* is initiated, allowing the student to use the device simulation. The tutor monitors the student in a rather simple way. It just observes which device subcycles the student steps through. If the student has stepped through most of the subcycles then the tutor marks as deleted the step that would continue the *Explain-Subcycles* procedure. Of course this is somewhat arbitrary since it is not clear how best to determine what a student learns from using a simulation. The key point is that if the tutor can detect that a student-initiated activity renders future activities redundant, the planner can eliminate them.

The basic curriculum planning scenario (eighth scenario) has been folded into the other scenarios. For any scenario a lesson length can be optionally specified and a sequence of lessons produced. The time management scenario in which there is not enough time to finish a lesson was discussed in Section 2.2.2. In the time management scenario in which the lesson will finish too early (scenario ten) the tutor replans to take advantage of the remaining time rather than waste it. The tutor adds activities to the end of the lesson to make use of the

remaining time. Currently, it just lets the student use the device exploration in the remaining time. Again the pedagogical approach is overly simple but sufficient to demonstrate replanning for this circumstance. Another approach - not implemented - is to move activities from the next lesson into this lesson if there is sufficient time to finish them.

In the incremental planning scenario the initial instructional plan is not fully elaborated since instructional procedures are not selected for each lesson plan step. Instead, only intended methods of instruction (called *activities*) are planned. Each can be implemented by different possible procedures. The procedure for a lesson plan step is selected when that step is reached. For example, a lesson plan step could specify assessment of the student's knowledge of lower hoist subcycles. Alternate procedures are available for the activity, such as a multiple-choice test, true-false test, or short-answer test. One of these is selected when the step is reached. In principle, more information is available and a better choice can be made than when the initial instructional plan is generated. However, the current implementation does not yet take advantage of this new information. The main difference between the two modes of planning is that plan *generation* is faster for the incremental planner compared to the full-elaboration planner. Plan *execution* starts sooner in the incremental case but is slowed down marginally during execution since each step requires extra time for procedure selection.

3. Integration into the Lower Hoist Tutor

This section describes how the tutor components are integrated with the planner in the Lower Hoist Tutor. The boxes shown in Figure 1-3 are actually only conceptual modules, not independent software modules. Instead, they are implemented as declarative data structures and routines that interpret and update them. A more precise description of the integration follows. The planner generates an instructional plan that is a sequence of procedures. One of these procedures is shown in Figure 3-1. Its execution is controlled by the planner, but the procedure itself draws upon the courseware and domain expert to generate text and graphics presented to the student through the student interface. If the procedure performs assessment then the text are questions and the student's answers determine how to update the student model. The instructional procedure calls upon the domain expert to compare the student's performance to ideal performance. The key point is that the planner itself does not have actions that deal with these auxiliary modules. Instead, it controls instructional procedures that interact with the modules.

Communication is primarily via shared data structures called *blackboards*. A procedure can update the student model by storing records in a blackboard. It

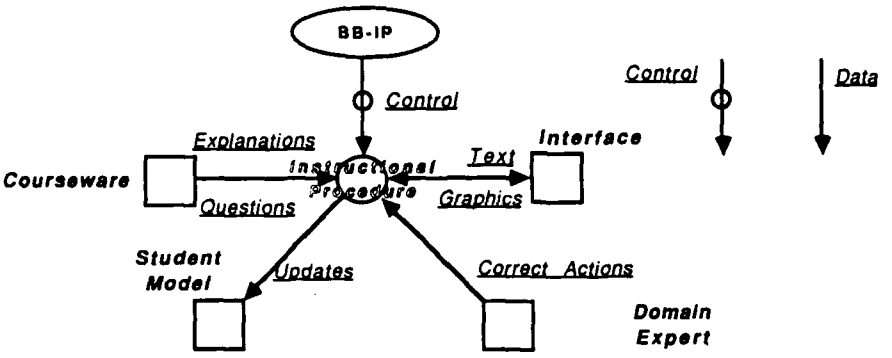


Figure 3-1: Direct control of auxiliary components by an instructional procedure

can emulate domain expertise by interpreting records of correct troubleshooting actions stored on a blackboard. The student interface is handled differently: an instructional procedure can issue graphics commands and text display commands directly and accept student responses, all without use of blackboards. Each of the non-planner tutor components shown in Figure 1-3 has a blackboard associated with it, except the interface. These are the *Device*, *Student Model*, and *Curriculum* blackboards described in Appendix C.1. Some components, such as the domain expert and student model, also have additional procedures to interpret or update the records on the blackboards.

These non-planner components have been implemented expeditiously since the focus of this research is the planner. No contributions have been made in student modeling, student diagnosis, curriculum design, or interface design. Difficult research issues in the design of these non-planner components are not addressed here. Instead, existing techniques used in other intelligent tutoring systems have been appropriated.

The student interface is STEAMER-based [Hollan 84]. Icons correspond to lower hoist parts. The same icon can be drawn differently to correspond to different part states. For example, in Figure A-1 the lower hoist rack piston UCK1 (labeled in Figure 1-5) is initially down (in photograph (h)) and then later up (in photograph (i)). Particular parts are highlighted by drawing a light blue box around them as shown in those two pictures. Device operation is shown by sequentially changing icon states to mirror the sequence of part states. A window to the left of the device simulation displays text. Pop-up menus, such as the one shown in Figure A-1 (a) allow the student to continue the lesson, ask a question, or make a request. Figure A-1 illustrates part of an instructional procedure that animates the device simulation while explaining the sequence of changes that

occurs. The student can also provide input by pointing to parts. For example, the student can ask what role a part plays by selecting a question template from a menu then pointing to the part, as shown in Figure A-2 (h) through (j).

Ideally, the domain expert should allow the tutor to solve troubleshooting cases and answer "What if?" questions the student might ask about the device. There is a causal model simulation of the lower hoist that can answer such questions, but it is not tied into the Lower Hoist tutor. Instead, troubleshooting expertise is simulated by interpreting a graph representing alternative traces of correct troubleshooting actions and states resulting from each action. These graphs are an offline compilation of correct troubleshooting obtained by interviewing our subject matter expert. The use of such a graph to monitor student troubleshooting performance is an approach borrowed from the SHERLOCK tutoring system [Lesgold 88].

The Lower Hoist Tutor's simulated domain expert includes a device knowledge base. This is a semantic network representation of lower hoist structural decomposition and operation. Figure 3-2 and Figure 3-3 show part of these representations. The representation of lower hoist operation is used to drive the device simulation when animating the lower hoist.

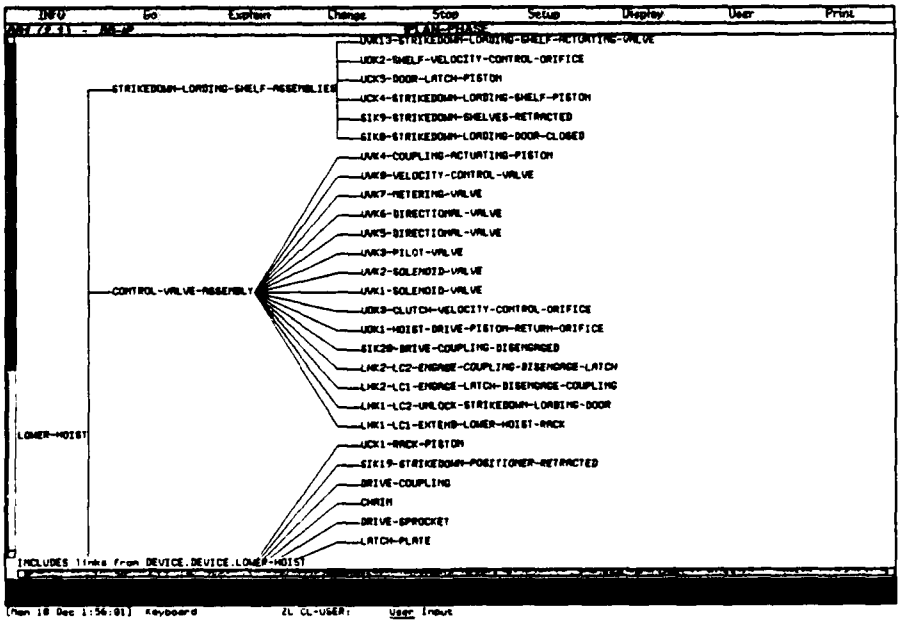


Figure 3-2: Lower hoist - structure

Some of the courseware consists of procedurally encoded domain-specific

instructional procedures. An example is the *Cycle-Overview* procedure which presents an overview of the subcycles of the lower hoist. However, most courseware is stored declaratively as blackboard records. These records are interpreted by domain-independent instructional procedures. For example, in giving a multiple choice test the questions for a domain-specific topic are part of these blackboard records.

The generic troubleshooting skills knowledge base is also part of the courseware. This is a representation of subskills required for effective troubleshooting of complex hydraulic-electronic-mechanical devices. It is shown in Figure 3-4 with vertical links representing prerequisites. Device structure and operation must be understood along with a general theory of troubleshooting. Other component skills include understanding possible faults of the device, being able to predict consequences of faults, and being able to reason from symptoms to possible underlying faults. This is a high level strategy for teaching troubleshooting of hydraulic-electronic-mechanical devices. It was derived from analyzing the lower hoist reference manuals and the subject matter expert's approach to teaching this material in the classroom. It is *not* based on any cognitive theory but is sufficient to test the instructional planner and produce plausible lesson plans.

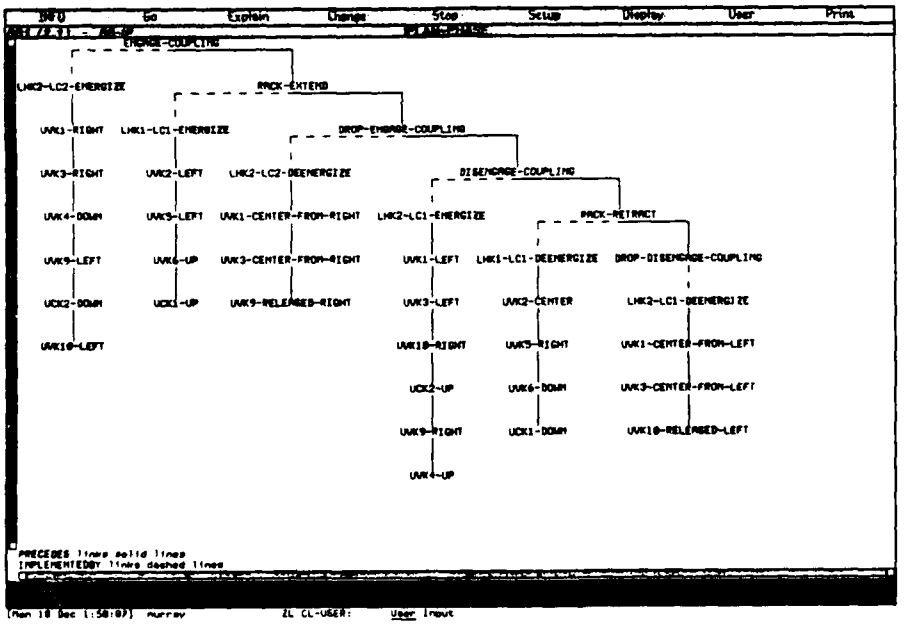


Figure 3-3: Lower hoist - operation

The student model consists of three parts:

1. *Cognitive stereotypes* - a list of inferred cognitive stereotypes [Rich 79] (e.g., a student could be described as HYDRAULICS-DEFICIENT, HAS-ELECTRONICS-PREREQUISITE, HAS-MECHANICS-PREREQUISITE.
2. *An inferred aptitude* - One of HIGH, AVERAGE, or LOW.
3. *An overlay of the domain-specific skills* - Associating certainty factors with domain-specific skills.

The first and second characterize the student's background knowledge and overall capability; the last characterizes his acquisition of device-specific troubleshooting skills.

The *cognitive stereotypes* indicate whether the student does or does not have the expected prerequisite background in hydraulics, electronics, and mechanics. For each background area *x* the student is assigned either the stereotype HAS-*x*-PREREQUISITE or *x*-DEFICIENT. The set of cognitive stereotypes are inferred from the pre-instruction questionnaire. If the student misses more than a certain number of the questions in an area such as hydraulics then he is inferred to be deficient in that area. The cognitive stereotypes are so named since they set up stereotypical expectations of student performance for the device-specific troubleshooting skills. For example, if the student is inferred to be HYDRAULICS-DEFICIENT then the tutor initially believes that the student does not understand how the UVK4 hydraulic valve operates.

The *aptitude* represents the tutor's classification of the student's overall learning and intellectual capability. It is inferred from the student's performance on all parts of the pre-instruction questionnaire and the student's educational background.

The student model *overlay* [Carr and Goldstein 77] represents the tutor's beliefs that the student has acquired each of the device specific troubleshooting skills. Associated with each node of the domain-specific skills is a pair of numbers. Each pair represents the tutor's belief that the student knows that skill and the tutor's certainty in its assessment. The tutor's belief is represented from -5 (student does not know skill based on evidence so far) to +5 (student knows the skill). The certainty of the tutor's belief is based on the amount of accumulated evidence so far. It ranges from 0 (no certainty) to 10 (very certain). Misconceptions are not explicitly represented in this student model.

During the course of the instructional plan various *assessment procedures* may test the student directly by asking questions, or indirectly by monitoring his performance on some task, such as troubleshooting. Each assessment procedure updates the student model based on the data it gathers. Presently each assessment

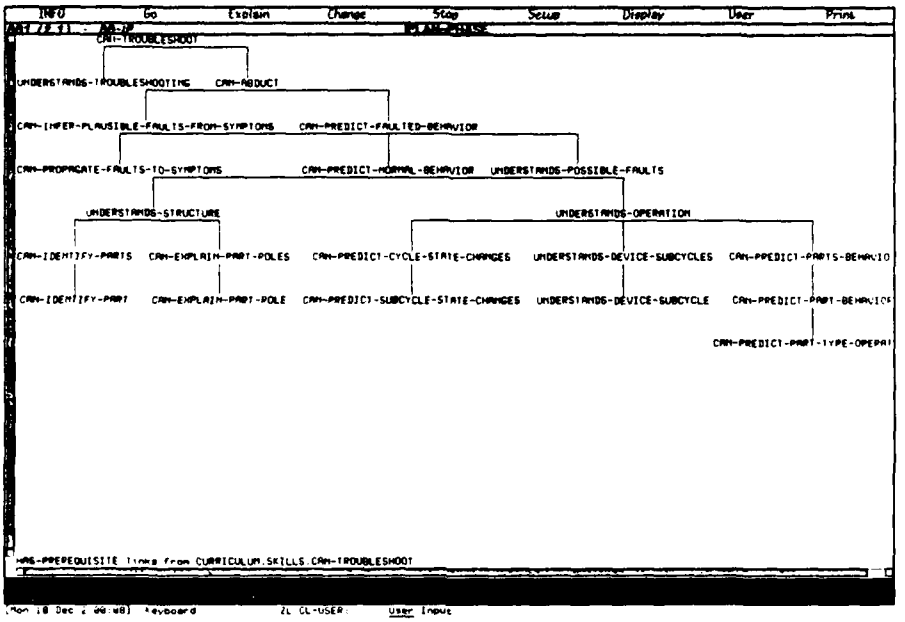


Figure 3-4: Generic skills for troubleshooting hydraulic-electronic-mechanical devices

directly updates only one node in the student model, and then effects are propagated to related nodes. Changes to a node affect its ancestors since the node represents one of their prerequisite skills. Propagation is fairly simple. It is based on approaches used in IMTS [Towne 89] and BIP [Barr, et al 76]. The value of a node is determined by combining direct evidence for that node with evidence for its immediate children (its prerequisites). More weight is given to direct evidence then evidence propagated upwards.

4. Plan Representation

This section presents the hierarchical instructional plan representation. There are three levels:

- *Instructional objectives* - Instructional objectives are the goals of the tutor. These are domain specific skills which the tutor intends for the student to acquire (e.g., CAN-PREDICT-NORMAL-BEHAVIOR-LOWER-HOIST).

- *Activities* - Activities are abstract methods for achieving these goals. The most common activities are to cover a topic (e.g., COVER-TOPIC-PART-ROLE-UVK4) or perform assessment for a skill (e.g., ASSESS-UNDERSTANDS-STRUCTURE-LOWER-HOIST).

- *Procedures* - Procedures are instructional routines that implement specific activities (e.g., the *Explain-Subcycles* procedure illustrated in Figure A-2).

Multiple procedures are available for the same activity. For example, different kinds of tests can be used for student assessment. Also, the same procedure may be used for different activities by setting its parameters differently.

Figure 4-1 shows a small instructional plan, for teaching the structure of the lower hoist, assuming that it only had two parts. The top level is the *instructional objectives level.* On it, the top-level instructional objective has been broken down into two subordinate (i.e., prerequisite) objectives. First, that the student can explain the roles of the parts of the lower hoist. Second, that the student understands how the different part types operate. These objectives are broken down still further. First, the student must understand the role of each of the two parts. Then since LHK1 is a solenoid assembly the student needs to understand how solenoids operate. Similarly, since UVK4 is a hydraulic valve the student needs to understand how hydraulic valves operate.

Now we consider the *activities level.* There is one activity for each terminal skill node plus additional activities to improve discourse flow and monitor plan progress. The activities COVER-TOPIC-ROLE-LHK1 and COVER-TOPIC-ROLE-UVK4 simply present text describing the role of the two lower hoist parts. The activity MOTIVATE-TOPIC-STRUCTURE-LOWER-HOIST has been added to improve discourse flow. It explains to the student that it is important to understand lower hoist structure in order to perform effective troubleshooting. The activities COVER-TOPIC-SOLENOID-OPERATION and COVER-TOPIC-VALVE-OPERATION present material explaining the operation of solenoids and valves. The final activity ASSESS-UNDERSTANDS-STRUCTURE-LOWER-HOIST tests the student to determine if the plan has had its intended effect.

The final level is the *procedures level.* For each activity a single procedure has been chosen to carry it out.[4] For the activities COVER-TOPIC-ROLE-UVK4 and COVER-TOPIC-VALVE-OPERATION alternate candidate procedures that were not selected are also shown. For example, to cover UVK4's role the tutor can use either *Part-Role* or *Demo-in-Cycle*. *Part-Role* presents a textual description of UVK4's role. It also highlights and labels it in the device simulation. *Demo-in-Cycle* demonstrates the part's operation in the lower hoist cycle by animating the device simulation. Similarly, the topic of hydraulic valve

---

[4]There is a one-to-one mapping between activities and selected procedures in the current plan representation. The use of multiple procedures for activities has not been explored.

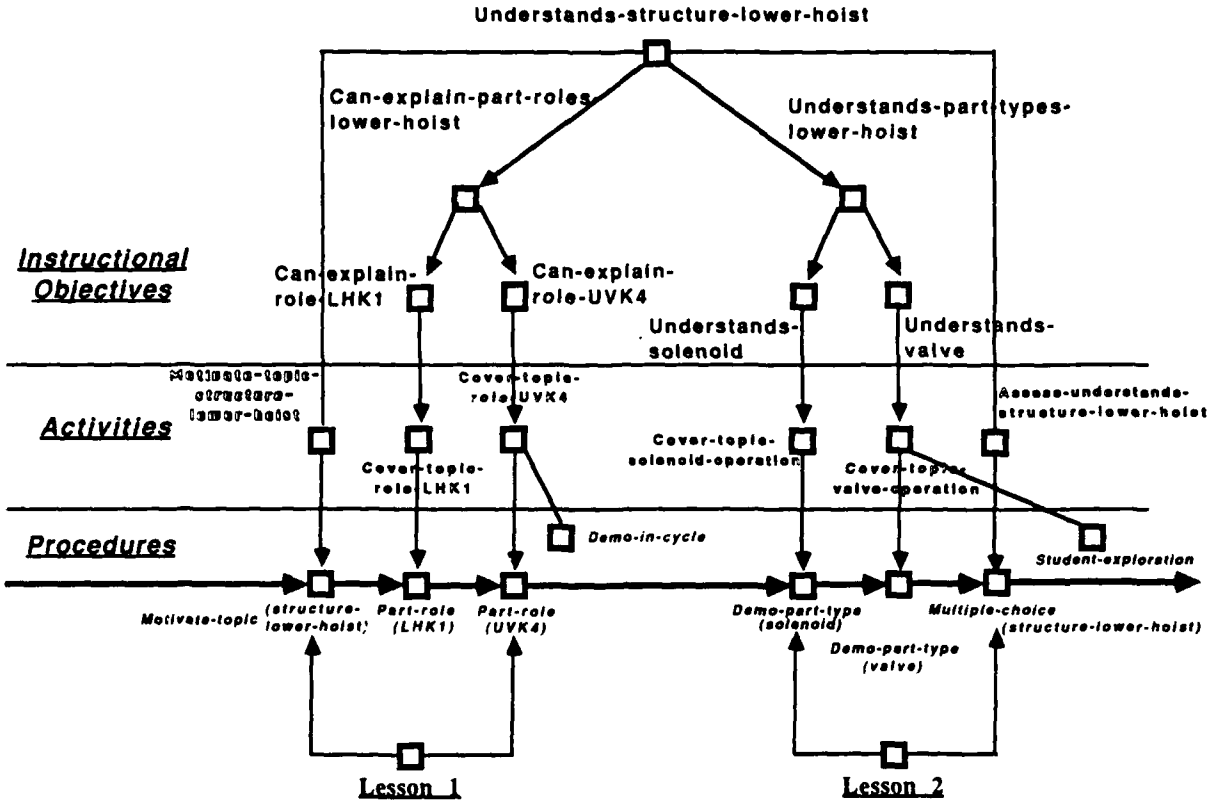


Figure 4-1: A simple instructional plan

operation can be taught either by *Student-Exploration* or by *Demo-Part-Type*. The first procedure asks the student to use the device simulation to explore the operation of a particular valve. The second demonstrates different valve states and how they affect connecting parts. Parameters, which are shown in parentheses, allow the same procedure to be used for multiple purposes. For instance, *Demo-Part-Type* is used for two different kinds of parts.

The procedures selected to carry out the activities are also linked together in a sequence and implicitly separated into individual lessons. Lesson boundaries are represented by pointers from the two *partition objects* at the bottom. Each has a pointer to the beginning and end of the lesson it represents. In the examples given in Section 2 it was only this bottom level that was shown. The previous simplified representation would show these two lessons as:

Lesson 1

Motivate-Topic[**structure lower hoist**]
Part-Role[**lhk1**]
Part-Role[**uvk4**]

Lesson 2

Demo-Part-Type[**solenoid**]
Demo-Part-Type[**valve**]
Multiple-Choice[**structure lower hoist**]

Parameters specify what parts, subcycles, or topics that a procedure should address. Many procedures operate in a data-directed manner. For example, the *Part-Role* procedure retrieves text discussing a part's role directly from the part's description in the device knowledge base. Similarly, *Explain-Subcycles* refers to the device knowledge base to retrieve text to explain the next part state change and to decide how to animate the device simulation.

Each procedure is broken down into steps called *procedure steps*. A procedure can be interrupted between procedure steps and then resumed later. Typically, a procedure step displays another paragraph of text, or makes an incremental change to the device simulation shown on the student interface. For assessment procedures, each procedure step asks a question. Procedures are written this way so the student can ask questions or make requests between procedure steps. Another reason is to monitor procedure execution and student progress since the tutor can assess progress or lack of progress between procedure steps. If progress is insufficient then the tutor can interrupt the procedure to adjust its parameters or abandon it altogether.

Procedures are the primitive actions in instructional plans since it is not cost-effective to apply control reasoning to reason about more primitive actions. Instead, more primitive actions, such as presenting a text paragraph, are performed within procedure steps. There is no need to incur the cost of another layer of interpretation using the blackboard architecture simply to emulate procedural control constructs for sequencing, looping, and conditional branching. The instructional plan becomes unnecessarily detailed and the planner is slowed down considerably when every procedure step is represented in the plan. Unnecessary overhead is introduced since planner knowledge sources are required to replicate simple procedural control constructs such as LISP's PROGN, COND, and DO.

The use of an intermediate level of plan abstraction allows incremental planning. The planner can intend to cover a sequence of topics without deciding

exactly how until those topics are reached. Although incremental planning has only been demonstrated in BB-IP - as opposed to being well developed and used to advantage - this intermediate level of representation supports it.

The reason that the instructional objectives level is important is that it supports replanning. BB-IP needs to know why it was doing a procedure to determine what to do if that procedure fails. This representation of plan rationale supports replanning when previously satisfied instructional objectives are no longer maintained, or when pending instructional objectives are discovered to be already satisfied.

5. Plan Generation, Execution, and Replanning

Figure 5-1 presents an overview of the operation of the Blackboard Instructional Planner. The generic skills knowledge base is instantiated with the device knowledge base to form a cartesian product of domain-specific skills. For example, if a generic skill is UNDERSTANDS-DEVICE-SUBCYCLE with parameter *subcycle* and if the device knowledge base has the six subcycles ENGAGE-COUPLING, EXTEND-RACK, DROP-ENGAGE-COUPLING, ENGAGE-LATCH, RETRACT-RACK, DROP-ENGAGE-LATCH then there will be six domain-specific skills that result:

1. UNDERSTANDS-DEVICE-SUBCYCLE-ENGAGE-COUPLING

2. UNDERSTANDS-DEVICE-SUBCYCLE-EXTEND-RACK

3. UNDERSTANDS-DEVICE-SUBCYCLE-DROP-ENGAGE-COUPLING

4. UNDERSTANDS-DEVICE-SUBCYCLE-ENGAGE-LATCH

5. UNDERSTANDS-DEVICE-SUBCYCLE-RETRACT-RACK

6. UNDERSTANDS-DEVICE-SUBCYCLE-DROP-ENGAGE-LATCH

The instructional plan is produced through a plan generation process that will be described shortly. The plan is shown in Figure 5-1 as if the lesson plans were disjoint, but they actually share goal substructure, as shown in the example plan of Figure 4-1. The instructional plan usually includes assessment activities. When these are executed the student model is updated. These changes, along with student questions and requests, can lead to replanning. Replanning causes *plan edits* which add or delete lesson plan steps, alter procedure parameters, or repartition the remaining instructional plan.

Figure 5-2 shows the resources used in plan generation. The activities

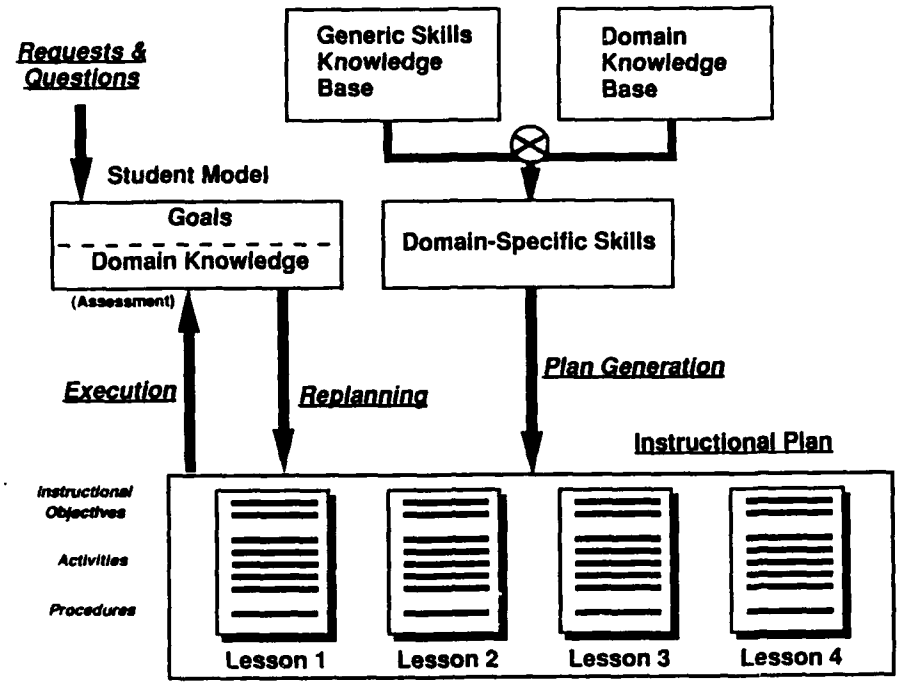


Figure 5-1: Overview of planning in the Blackboard Instructional Planner

library is a set of generic activities (e.g., COVER-TOPIC, ASSESS-TOPIC, DEMO-SKILL) that can be used to achieve objectives. The procedures library is a set of instructional routines that can be used to achieve activities (e.g., EXPLAIN-SUBCYCLES). These libraries are part of the courseware for the domain, although most of the activities and procedures can apply to other similar domains. The student model is a resource since it is used to filter out those objectives, activities, and procedures that are inappropriate for the student.

Another resource is the library of *discourse critics*. These can be thought of as demons or special purpose rules that improve the discourse flow or coherence of the instructional plan. They critique and revise the instructional plan either during plan generation or immediately after plan editing. For example, one discourse critic detects abrupt topic transitions and adds a call to the *Transition* procedure to signal the context change in the tutor's presentation. The operation of this discourse critic was illustrated at the end of Section 2.2.1 and in Figure 2-1.

Once the instructional plan has been developed, it is partitioned by the *lesson partitioning algorithm*. This algorithm estimates lesson length and the abruptness of transitions to decide where to place partitions between lessons. It

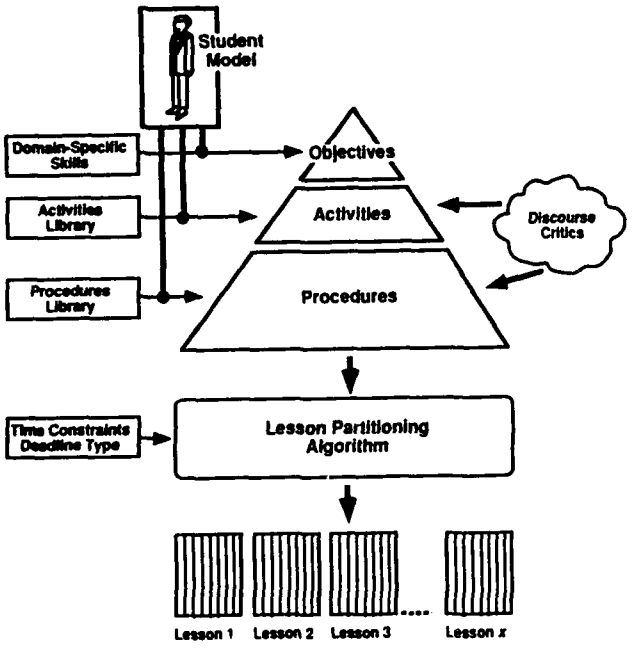


Figure 5-2: Plan generation

first estimates the time required for each procedure. Associated with each procedure is a function to estimate the number of procedure steps and a second function to estimate the average time per step. These are used to arrive at the overall estimate. The partitioning algorithm also takes into account the disruption in the semantic coherence of the instruction caused by placing a partition between two procedures. For example, ending a lesson with a procedure that provides motivation for a topic in the next lesson is less desirable than having the motivation immediately precede the topic motivated.

A heuristic function that combines these two criteria - semantic coherence and deviation from target lesson length - is used to score how bad different possible partition placements would be. The algorithm is quite simple. It moves a possible partition placement from the beginning of the lesson towards the end. The measure of how bad the partitioning is decreases until a minimum is reached and then the partition is placed. The procedure repeats with the remaining lessons. This is a simple hill-climbing procedure and obviously more sophisticated scheduling or search algorithms could be applied.

Plan generation and execution occur in phases called *control phases*. These are states where only certain kinds of planning actions are performed. BB-IP's

control phases are shown in Figure 5-3. Phases on horizontal lines are next to the planning level they affect. Planning actions that occur in these phases either add to or modify a partially refined instructional plan. The special *Meta-Level* phase decides what plan phase to jump to next.

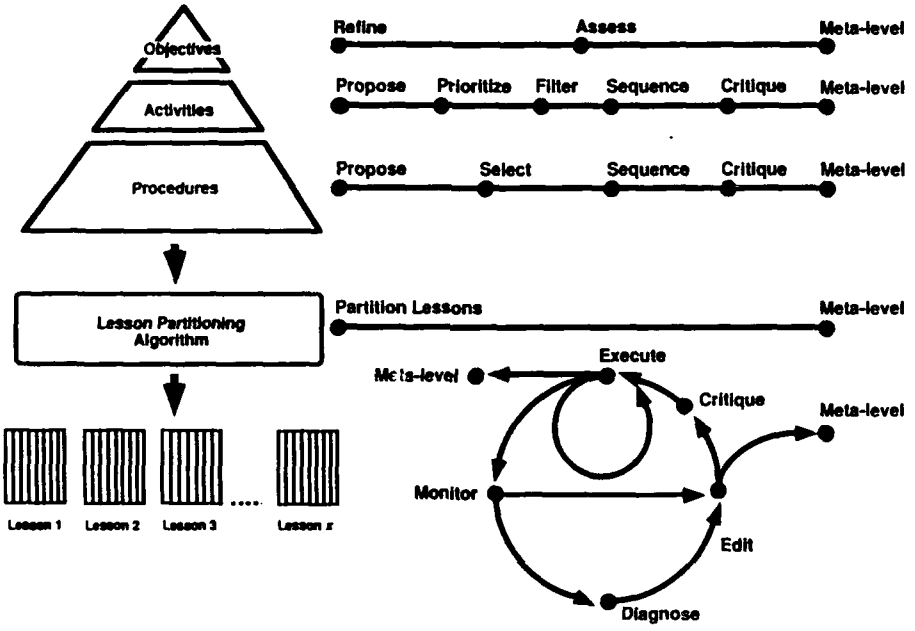


Figure 5-3: Control phases in plan generation and execution

Consider plan generation first, assuming complete top-down plan elaboration. The basic sequence of phases is:

1. *Refine Objectives*- The top-level instructional objective is refined by copying prerequisite objectives from the domain specific skills.
2. *Assess Objectives* - The student is given a pre-instruction questionnaire to initialize the student model.
3. *Propose Activities* - Activities to achieve the objectives are proposed.
4. *Prioritize Activities* - Numerical priorities are assigned to each activity to indicate its importance.
5. *Filter Activities* - Only those activities above a threshold are retained.
6. *Sequence Activities* - These activities are now sequenced.

7. *Critique Activities* - Improvements in the discourse flow are made.
8. *Propose Actions* - Candidate procedures for each activity are proposed.
9. *Select Actions* - Heuristics are used to select the best procedure for each activity. Parameters are set for the procedures.
10. *Sequence Actions* - Procedures are sequenced in the same order as their parent activities.
11. *Critique Actions* - The plan is again critiqued and improved.
12. *Partition Lessons* - Partitions are laid down, breaking the instructional plan up into lesson plans.

The *Meta-Level* phases at the end of each horizontal line in Figure 5-3 were not mentioned since they just produce the sequencing above. However, when incremental planning occurs phases 8, 9, 10, and 11 are skipped.

The planner uses the student model to customize the plan in several of the phases above. In the *Prioritize Activities* phase the priorities for covering various topics and performing other activities are set based on the student model. Similarly, in the *Select Actions* phase the selection and setting of procedure parameters also depend on the student model.

After the *Partition Lessons* phase execution begins. The *Meta-Level* phase moves the planner into the execution loop at the bottom of Figure 5-3, starting with the *Execute* phase. The planner stays in the *Execute* phase as long as the instructional plan is executing smoothly. Interruptions occur when the student model or time do not change as expected or if there are student-initiated questions or requests. The planner enters the *Monitor* phase and the reason for the plan's interruption is recorded as a *plan execution complaint.*

A question or a request is deferred if it will be handled later in the instructional plan anyway. Otherwise it is granted immediately. In the former case the tutor explains why the request is being deferred and continues with the *Execute* phase. In the latter case a step performing the procedure that answers the question or grants the request is spliced into the lesson plan in the *Edit* phase. The *Critique* phase adds a transition step then execution resumes. The next step to be executed is the step satisfying the student's request.

An unexpected change to time remaining or the student model means that the plan has some flaw. The *Diagnose* phase attempts to infer what is wrong. This phase ends when a decision has been made about what the problem is. Then the *Edit* phase either adds or deletes lesson plan steps, or alters procedure

parameters, according to the kind of problem. The steps added may only be at the activity level, requiring a digression back to the plan generation process to select procedures. Then this new plan patch is critiqued and execution resumes.

Several planning actions may occur in any one control phase, although abstractly the phases can be considered as one planning action. For example, the *Propose Activities* plan phase is actually broken down into three separate planning actions. One proposes activities that cover topics. Another proposes assessment activities that monitor plan progress. The last proposes pedagogical activities such as letting the student explore the device exploration or practice new skills. Similarly, the *Execute* phase in the execution cycle may correspond to the execution of many individual procedures and procedure steps. Most of the time the planner is in the *Execute* phase.

Changes to plans are called *plan edits* or *plan patches*. These involve adding steps, deleting steps, adjusting parameters, or repartitioning lessons. Objectives are not added or deleted but their status as satisfied or unsatisfied can be altered which has the same effect. Another approach not explored in the current planner is to discard the remaining lesson plan or even the remaining instructional plan and plan anew from the current situation. This approach could be disruptive to the student's expectations and wasteful in planning time. However, this may be the best approach in some situations that would otherwise require numerous interacting plan edits.

Planning and execution are interleaved through the use of control phases and in particular the *Meta-level* control phase. The *Meta-level* phase allows suspending the execution cycle to return to the generation process and then resume execution. This interleaving is used in three ways in the current planner (all implemented):

1. *Incremental Planning* - execution of the plan, refined only to the activity level, is interleaved with selection of procedures and critiquing the refined plan.
2. *Elaborating and Integrating Plan Patches* - plan patches, consisting of only activities, are refined to the procedure level and then spliced into the current plan before execution resumes.
3. *Initial Assessment* - a single-step plan for performing assessment is generated and executed to assess the student's knowledge of the instructional objectives. This is a digression that is initiated by the *Assess Objectives* plan phase. Although not currently performed, the planner could develop more elaborate multi-step assessment plans in this manner.

6. The Blackboard Architecture and its Role

This section describes how the blackboard software architecture has been used to implement the Blackboard Instructional Planner. The role of the architecture and its implications for implementing future ITS systems are discussed. First we clarify what we mean by "architecture" since this term can be used in many ways.

A software *architecture* is defined here as a layered software environment that defines a knowledge base structure and inference mechanism. It is a general purpose inference engine or reasoning subsystem [Feigenbaum 88] that can be implemented in any standard programming language, with varying degrees of efficiency. The architecture defines a new programming language at a higher level [Chandrasekaran 89] than those used to implement it. The architecture is not the programming language itself, rather it is a virtual machine that can interpret this high-level programming language. Associated with a software architecture are a set of programming conventions for using it and a set of problems that it is most *appropriate* for. Other problems can be handled because of Turing compatibility, but part of what differentiates one architecture from another is the target class of problems it is designed to handle best. To simplify the problem-solving for the intended class of problems the architecture imposes constraints on the user. Some of these are explicit - in terms of constructs provided by the higher-level programming language - and some are implicit - in terms of the programming style associated with the architecture. In turn the virtual machine defined by the architecture is specially designed to accelerate and structure the problem-solving process by taking advantage of these constraints. Additionally the structure imposed may simplify human conceptualization of the problem and its solution, independent of its use by the architecture's implementation.

Examples of software architectures are production rule languages, neural networks, blackboards (discussed below), and logic programming. Object-oriented programming would not be a software architecture by this definition since no virtual machine, knowledge base, or inference engine is provided. Instead, object-oriented programming is a programming style and a set of tools that can be used to build software architectures. If we allow any change to a data structure to be considered an inference, and any program to define a virtual machine, then all programs define architectures and the term is not useful. The inference engine defined must be general purpose.

There is a looser definition of architecture that can easily be confused with the stronger definition given above. Consider the diagram in Figure 1-3. It is a *functional diagram*, showing conceptual layout, rather than an architecture since

it does not meet the criteria above. Although it is intended for a class of problems (intelligent tutoring systems) it does not define a higher-level programming language or a virtual machine that interprets this language. If the modules corresponded to separate processes and the arrows to message passing then the diagram would correspond to a looser definition of architecture. This looser definition of architecture specifies a software system's organization by defining its constituent software modules and a static set of data and control flow pathways. Message passing conventions may also be defined. This looser definition is more a software infrastructure rather than a higher-level virtual machine. This paper uses the first definition of architecture unless otherwise specified.

There are three defining features of the blackboard architecture:

1. *Hierarchically structured global database* - this is the blackboard, typically organized into levels to reflect the hierarchy.[5] The evolving problem solution is recorded here as objects on the levels. The objects can have attributes and be linked to other objects.

2. *Independent knowledge sources* - these are software modules that communicate only by accessing and recording data on the blackboard.

3. *Agenda control mechanism* - the knowledge sources that can be run are stored on an agenda. Problem-solving heuristics are used to select which knowledge source to run next.

The programming conventions for using blackboards effectively are:

- *Knowledge source independence* - the knowledge sources should not make assumptions about the presence of other knowledge sources or their sequence of execution. They cannot communicate by global variables or message passing, only by the blackboard.

- *Knowledge source granularity* - the knowledge sources make significant contributions to problem solving. The contribution should be large enough to justify the overhead of the agenda control mechanism. The granularity of knowledge sources is typically much greater than that of production rules in production rule languages.

- *Problem structuring* - the problem is represented on the blackboard at various levels of abstraction.

---

[5]BB1 [Hayes-Roth 84] is as described here; GBB [Corkill 87] allows more complicated structural breakdowns of the blackboard into spaces of n-dimensions.

6.1. Overview of the Implementation

BB-IP is built on the BB1 Blackboard Architecture [Hayes-Roth 84], which allows multiple blackboards, rather than just one. These are the main blackboards used by BB-IP:

- *Instructional Plan* - represents the three-level instructional plan. As discussed before, the levels are:
  1. *Instructional Objectives*
  2. *Activities*
  3. *Procedures*
- *Student Model* - represents the student model.
- *Device* - represents the structure and operation of the device.
- *Planner Control* - represents meta-level decisions such as whether to continue planning or start execution. Problems noted in the plan's execution and decisions about their cause are also recorded.

The device blackboard is not updated during execution; the others are.

The knowledge sources of BB-IP are planning actions. They primarily modify the *Instructional Plan* blackboard. The kinds of actions performed are:

- *Plan generation* - new objects are added to the instructional plan and linked to the objects they refine. These objects represent objectives, activities, or procedures.
- *Plan execution* - a step in a procedure is executed.
- *Plan monitoring* - some problem in plan execution is noted. For example, the time available or student model is not as it should be.
- *Plan diagnosis* - a decision is made regarding a plan failure.
- *Plan revision* - part of the plan is modified. New activities or procedures may be added or others deleted.

The execution cycle of BB-IP is shown in Figure 6-1. Changes to the blackboards are caused either by the execution of planning actions, or by student interaction. These changes are called *events*. They trigger *knowledge sources* (*KSs*) creating activation records which are placed on the agenda. These *knowledge source activation records* or *KSARs* record the knowledge sources and their variable bindings when triggered. The *scheduler* (described below) selects the next KSAR to run. Then the next KSAR is interpreted. Its execution may cause changes to the student interface, the instructional plan, or both. This cycle continues until the instructional plan has been completed.

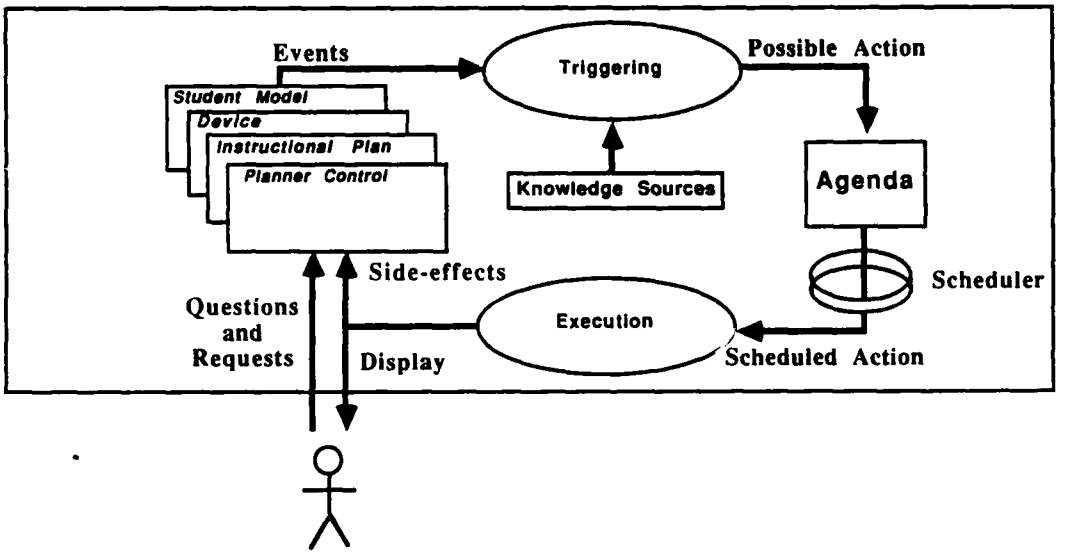


Figure 6-1: Execution cycle of the Blackboard Instructional Planner

Control reasoning is reasoning about what to do next. In BB-IP it is quite simple. Triggering of knowledge sources is restricted not only by the events which have just occurred but also by the current control phase, which is recorded on the *Planner Control* blackboard. The special *Meta-Level* knowledge source switches control phases when appropriate. Several KSARs may be on the agenda during each control phase. Each KSAR has an attribute called its *priority* or a function that can be called to compute this number. The scheduler always selects the KSAR with highest priority to execute next.

Further details of the implementation are provided in the appendices. Appendix C describes all the blackboards and knowledge sources. The first version of the planner, which relies more heavily on BB1's built-in control reasoning capabilities, is described briefly in Section 7 and more extensively in Appendix F.

6.2. Support for Dynamic Instructional Planning

The blackboard architecture facilitates the implementation of dynamic instructional planners by supporting an explicit plan representation, meta-level reasoning, interruptibility, and rapid prototyping. Other reasons cited [Hayes-Roth 87a] for using blackboard architectures such as integrating

uncertain data, island driving,[6] and providing opportunistic control appear more relevant to interpretation applications (e.g., HEARSAY-II [Erman, et al 80]) than this planning application. If more sophisticated student modelling were performed these reasons might justify using a blackboard for constructing the student model.

An explicit instructional plan representation supports dynamic planning by representing plan rationale, future unexecuted actions, global resource allocation, and constraints on partially refined plans. The use of plan critics are also supported. These are explained in more detail below:

- *Plan rationale* - the instructional objectives represent the rationale for each plan step. Representing plan rationale supports replanning when current goals fail, previously satisfied goals are not maintained, or future goals unexpectedly become true. By knowing why steps are in the plan, BB-IP can detect when they are no longer necessary or what remains to be done if the steps do not have their expected effects.
- *Future unexecuted actions* - by representing future actions BB-IP can provide the student with lesson and course overviews, and plan smooth transitions back from interruptions to the main lesson plan.
- *Global resource allocation* - with an explicit representation of the overall plan BB-IP can appropriately allocate resources to best use its resources. Otherwise local decisions made now could deplete resources required for future goals.
- *Partially refined plans* - An explicit representation of partially refined plans allows constraints on future plan steps to be represented without committing to a specific step; this supports the kind of incremental planning that BB-IP performs.
- *Plan critics* - can be used to improve initial plans. They detect interactions between different parts of the plans and make local optimizations by examining subsequences of plan steps.

BB-IP does not fully exploit these features although all are used. For example, only time is reasoned about in resource allocation. Similarly, BB-IP's incremental planning capabilities do not take advantage of additional information that can be gathered by deferred decision-making.

The blackboard architecture supports an explicit plan representation by

---

[6]Island driving focuses problem-solving activity on merging together those solution fragments that have the greatest support from the evidence available.

providing the blackboard data structure and mechanisms for defining blackboard objects, attributes, and links. Objects on blackboard levels can represent steps on different levels of plan abstraction. Links can represent step sequencing and refinement. Attributes can represent constraints or allocated resources.

Meta-level reasoning supports dynamic planning by allowing plan generation, refinement, execution, monitoring, diagnosis, and repair to be interleaved as required. It supports efficiency by focussing the planner's activity. The blackboard architecture supports meta-level reasoning by allowing control decisions to be explicitly represented on blackboards and by allowing these control decisions to affect the scheduling of knowledge sources. The scheduler and agenda control mechanism allow the planner to choose the best action (KSAR) possible given its current planning goals. These planning goals might be conflicting or diverse; the scheduler can choose that action that best contributes to achieving the most important goals.

Interruptibility is required by a dynamic instructional planner to allow unexpected student-initiated questions and requests. The blackboard architecture supports interruptibility by allowing control to switch from one action (KSAR execution) to another in different cycles. For example, in BB-IP the *Execute Procedure* knowledge source is used to execute the next step in an instructional procedure. It may need to be called many times to finish the procedure. If a student question occurs during the procedure another knowledge source to handle the question is triggered. That knowledge source has higher priority than *Execute Procedure* so it will execute next. In this way instructional procedures are interruptible between procedure steps.

Rapid prototyping facilitates construction of planners such as BB-IP and exploring different planning approaches with them. The planner has evolved incrementally. Increasing capabilities were added that required extending and revising the knowledge sources and plan representation. First, plan generation and execution were implemented but no lesson partitioning, plan monitoring, or dynamic replanning was performed. Next, plan monitoring and replanning for failed lesson objectives were added. These extensions required adding knowledge sources to monitor changes to the student model, to diagnose the cause of plan failure, and to edit the plan to fix the problem. The ability to handle student-initiated questions and requests was added next. A means of recording these interactions was required along with new knowledge sources for handling requests. Additional plan critic knowledge sources ensured that the tutor provided a smooth transition from student-initiated digressions back to the main lesson topic. When curriculum planning was added the instructional plan representation had to be extended to represent lesson plan partitions. A new

knowledge source to partition lessons was added at the same time. All of these changes were facilitated by the rapid prototyping environment afforded by the BB1 blackboard architecture [Hayes-Roth 84] that was used to implement BB-IP.

Blackboard architectures support prototyping by allowing knowledge sources to be selectively enabled or disabled. Other knowledge sources do not need to be modified if each has been written to be independent and to communicate only through blackboards. BB1 additionally provides graphic display of blackboards, events, and the agenda. KSAR execution can be traced and the display updated after each execution cycle. BB1 also provides windows with menu commands that allow blackboards and knowledge sources to be easily created or modified.

Each of these features - explicit plan representation, meta-level reasoning, interruptibility, and rapid prototyping - were essential to the BB-IP implementations (both planner versions) and *facilitated* by the BB1 blackboard architecture. But BB1 and blackboard architectures are not *required*. For example, MRS [Russell 85] would support many of the features listed except rapid prototyping. Alternatively, the planner shell SIPE [Wilkins 84] could have been extended, as will be discussed in Section 7. Finally, BB-IP could have been written from scratch in LISP. All could have been done but would have been more difficult.

The role of the architecture is to *facilitate* the solution to the problem, not to solve it. The architecture alone does not solve the problem; it is just a tool. The two implementations of the planner show that even one architecture (BB1) can be used in two quite different ways to solve the same problem. Focussing too much on the architecture can obscure the difficult task of determining what knowledge to add and how to structure it. Control issues about how to apply that knowledge must also be addressed. For example, the architecture may provide a meta-level reasoning capability, but the system designer must still determine how to use it and what meta-level knowledge to add.

7. Related Work

This section discusses related work in intelligent tutoring systems, planning, explanation generation, and blackboard applications. We start with a discussion of different approaches to control for intelligent tutoring systems, organized around the models of instruction shown earlier in Figure 1-2. A more detailed discussion, organized by control mechanisms rather than instructional approaches, can be found in [Murray 89b].

7.1. Related Research in Control for Intelligent Tutoring Systems

As mentioned earlier, most intelligent tutoring systems rely primarily on opportunistic tutoring strategies. The simplest approach, requiring no planning capabilities, is to provide a reactive learning environment consisting of a simulated microworld. The student discovers concepts and acquires new skills in the process of exploring the microworld. Examples of such instructional systems include STEAMER [Hollan 84], REFRACT [Reimann 89], SMITHTOWN [Shute 86] (considering just the economic simulation of the town), and LOGO [Papert 80] (since a graphic turtle simulates the motions of a robot turtle). Such environments may lack a student model, non-simulation domain expertise, and provide only limited or no tutorial interventions. They are discussed here since they rely heavily upon artificial intelligence techniques and environments. Learning environments currently being developed that exploit hypermedia may also adopt a similar discovery learning approach. Again, no planning is required.

Planning is also not required for tutors that act as coaches (e.g., WEST [Burton and Brown 79] or WUSOR [Stansfield 76]). These systems more closely fit the usual model of an intelligent tutoring system since they have domain problem-solving capabilities and a student model. The domain expert allows the tutor to compare the student's moves to an expert's to infer the student model. These tutors are still primarily reactive since they do not plan lesson content and delivery, or engage in extended discourse with the student. Instead, simple hints are provided when appropriate.

More complicated control mechanisms are required to handle mixed-initiative instruction. Discourse management techniques are required to allow the student to ask questions and make requests of a tutor using a Socratic method style of instruction. SCHOLAR [Carbonell 70], WHY [Stevens and Collins 77], MENO-TUTOR [Woolf, et al 84], and GUIDON [Clancey 79] are examples of such tutors.

A rule-based approach to dialog control is used in SCHOLAR and WHY. GUIDON builds on this work and introduces domain-independent tutorial rules, called *t-rules*, that handle discourse and changes to the student model. Each T-rule is a pre-stored plan consisting of a sequence of steps. Execution of each step can depend on the student model and dialog history. Step execution can cause text presentations, student model updates, or the execution of other t-rules [Clancey 87]. Overall dialog control is provided by a *discourse management network*; this network is a state transition graph of dialog states. Dialog states focus the discussion on domain rules, goals, data, or student hypotheses. Transitions between states are again managed by t-rules. So although GUIDON uses a discourse management network, its principal control mechanism is its t-

rules [Murray 88a].

In contrast, the principal control mechanism of MENO-TUTOR is its discourse management network. Rules associated with each dialog state determine discourse actions and a default next state. Specific meta-rules can override these defaults. MENO-TUTOR differs from GUIDON in its reliance on this discourse management network for control, and in its hierarchical representation of tutorial discourse decisions.

The key point is that these tutors are not just reactive, they have more sophisticated control mechanisms for local planning of discourse. The tutors plan only in the sense that they select a sequence of actions to perform. They do not plan in the sense of reasoning about the results of action sequences or performing global resource allocation among plan actions. Instead, control in GUIDON and MENO-TUTOR more closely resembles planners that perform hierarchical plan expansion with dynamic selection of pre-stored skeletal plans.

Discourse management techniques are not required for tutors that act as intelligent problem-solving monitors, although they are sometimes employed. Such tutors adopt a case-method style of instruction that is currently the predominant paradigm for intelligent tutoring systems. Socratic-style instruction is possible but most case-method tutors do not support sophisticated discourse management, with GUIDON being a notable exception. More commonly, as the student solves some problem the tutor assists the student with hints and provides the next step as a last resort. It also looks for and acts on opportunities to either introduce new material or point out possible misconceptions. Such tutors usually have sophisticated problem-solving capabilities that allow them to monitor the student's progress (examples include GREATERP [Reiser 85], RBT [Woolf 87], SOPHIE [Brown 75] and IMTS [Towne 89]) or to evaluate the final solution (examples include PROUST [Johnson 85] and TALUS [Murray 88b]).

Some of these tutors dynamically select problems to suit a student's skill level but many do not. BIP [Barr, et al 76], GREATERP [Reiser 85], and IMTS [Towne 89] belong in the former category. In contrast, PROUST and GUIDON assume that the case to discuss is pre-selected. Those systems that dynamically select a sequence of cases perform a simple kind of incremental planning of lesson content. However, these systems still lack an ability to plan and deliver customized expository instruction; instead they are intended to augment prior or concurrent instruction from the classroom (e.g., PROUST) or a workbook (e.g., GREATERP).

In summary, the opportunistic tutors discussed above use fairly simple control mechanisms:

- *Reactive planning* - as in SOPHIE-I or WEST.
- *Dynamic task selection* - as in BIP.
- *Dynamic selection and expansion of pre-stored skeletal plans* - as in GUIDON (using t-rules) or MENO-TUTOR (using a discourse management network).

Other possible control mechanisms for opportunistic tutors such as agendas are considered in more detail in [Murray 89c].

All of these control mechanisms are not dynamic instructional planners by the definition on page 2 since these control mechanisms do not

- *Construct a data structure representing the tutor's instructional plan* - Instead there is either a pre-stored plan, no plan at all, or else some other pre-stored data structure that is interpreted to determine course content and delivery. For example, GUIDON refers to a pre-stored MYCIN trace to select topics to discuss.
- *Interpret this explicit plan representation* - to control delivery of instruction.
- *Revise the plan* - as the tutorial situtation changes.
- *Perform global resource allocation* - because unlimited time is usually assumed.

Now we consider intelligent tutoring systems that are dynamic instructional planners, according to the definition on page 2. They have the potential for integrating the opportunistic teaching methods presented above with the ability to generate customized expository instruction, as shown in Figure 7-1. BB-IP is a step in this direction, but much more work remains to be done. We consider other approaches to planner-based control to tutoring systems below, along with earlier planners and planner architectures implemented in this project. Differences between these systems and BB-IP are discussed in some detail to help place the current work in perspective and trace the evolution of planners developed in this project.

The first major step in the direction of a dynamic instructional planner was taken by Peachey and McCalla [Peachey 86] in their STRIPS-based lesson planner. The planner's operators are very high level: they deliver instruction and perform assessment for topics in economics. The plan generated is non-hierarchical but explicitly represents expected changes to the student model. Replanning occurs when these changes do not occur and none of the contigent branches in the plan apply. A plan editor determines a plan patch to change the student model from what it is to what it should be before the remainder of the

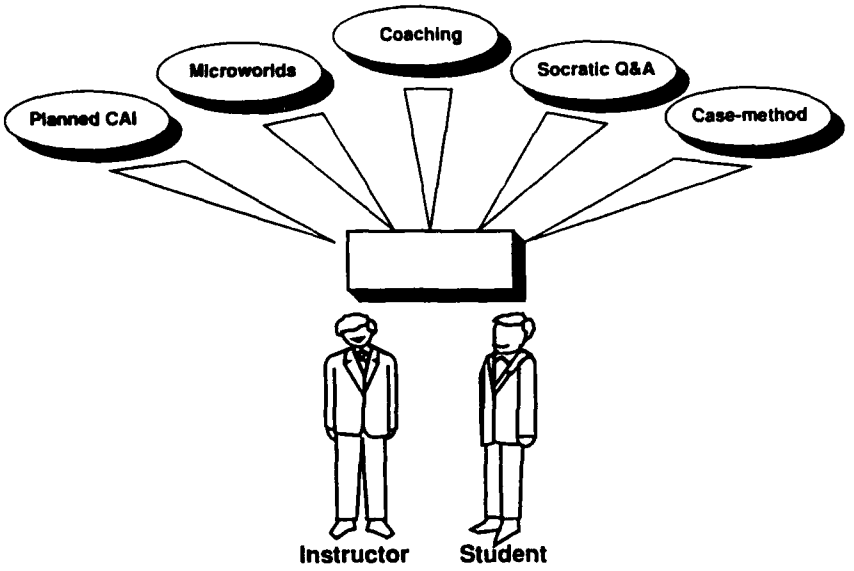


Figure 7-1: Integrating plan-based and opportunistic teaching paradigms

plan can be continued.

BB-IP differs in many ways, although there are some similarities. Unlike the planner above, the plan representation of BB-IP is hierarchical, includes time estimates for plan steps, but does not have conditional branches. It does not have to be generated all at once, instead it can be refined incrementally. The Peachey and McCalla planner, like BB-IP, can generate customized lesson plans and adaptively replan according to changes in the student model. But it does not address mixed-initiative instruction, time management, or detailed lesson delivery. The emphasis is on planning lesson content. The planner's actions are very coarse grained (e.g., TEACH-COD for teach concept of demand) and assume that all changes to the student model are known after each action is executed. Thus each action is also responsible for assessment of the concept taught. In contrast, BB-IP unbundles presentation and assessment by having separate instructional procedures for each. It plans how and when to perform assessment. To support mixed-initiative instruction, BB-IP's procedures are interruptible. Finally, BB-IP plans the amount of time per lesson and adaptively replans as time available changes; the STRIPS-based planner does neither.

IDE-INTERPRETER [Russell 88a] is a dynamic instructional planner that is more similar to BB-IP than the Peachey and McCalla planner. IDE-INTERPRETER delivers instruction for the IDE [Russell 88b] (instructional design environment) system. The planner interprets a set of rules to expand

instructional goals into subgoals. IDE-INTERPRETER and BB-IP are similar since the planning approach of both is basically top-down plan expansion, both represent the goal structure in lesson plans to support incremental planning and adaptive replanning, and both execute instructional routines (called *instructional units* in IDE-INTERPRETER) as primitive plan actions. During each cycle of IDE-INTERPRETER's execution it selects and executes an instructional unit relevant to the current instructional goal. It does not select these before it begins execution, so its operation is similar to BB-IP's incremental planning approach.

IDE-INTERPRETER's instructional units are similar to BB-IP's instructional procedures, but are not broken down into smaller steps to support interruptibility and monitoring. It does not appear that IDE-INTERPRETER sets the parameters of its instructional units as BB-IP does, but instead IDE-INTERPRETER selects the most appropriate instructional unit available. The student model is not updated until after the execution of an instructional unit, at which time a history list of interactions is analyzed. Student requests that could not be handled by an instructional unit would also be handled at that time. In contrast, BB-IP monitors the execution of instructional procedures, updates the student model, and handles student input during procedure execution (between procedure steps). Thus, BB-IP deals with issues of procedure monitoring, interruption, suspension, resumption, abandonment, and the planning of smooth transitions to and from student-initiated digressions that occur during procedure execution; IDE-INTERPRETER does not address these issues. For example, BB-IP monitors procedures and adjusts procedure parameters between procedure steps if student performance is too low; IDE-INTERPRETER cannot make such adjustments.

BB-IP also has greater control reasoning capabilities; these are facilitated by its explicit representation of control decisions separate from the instructional plan. Its use of control phases allows it to deviate from plan generation, execution, and refinement to reason about causes of plan failure and the best plan edits to fix the problems. Encoding these kinds of deviations appears more difficult in IDE-INTERPRETER since it does not explicitly represent control decisions.

There is insufficient space to cover all the related work in instructional planning for intelligent tutoring systems. Research in this area is ongoing at the University of Amsterdam (EUROHELP project [Breuker 87]), University of Saskatchewan (SCENT project [Brecht 89]), and Florida State University (TAPS intelligent tutoring system [Derry 88]).

Now we consider two precursors to BB-IP developed under this same research contract. BB-IP's immediate predecessor is the *first* version of the Blackboard Instructional Planner, described in [Murray 89b] and [Murray 89a].

This first version will be referred to as BB-IP-1 and the second (current) version as BB-IP-2 or just BB-IP when there is no chance of confusion. Preceding both versions of the Blackboard Instructional Planner is the research on planner and ITS architectures by MacMillan described in [MacMillan 87] and [MacMillan 88]; this will be discussed first.

MacMillan's research focuses on the design and implementation of planner and ITS *architectures*, rather than the implementation of any specific dynamic instructional planner. In [MacMillan 87] two architectures are presented:

1. *The BLACKBOARD-instructor* - an ITS architecture.
2. *The SIIP architecture* - an architecture for dynamic instructional planners, to be used in conjunction with the BLACKBOARD-instructor.

The SIIP architecture is an architecture in the first sense defined on page 35, i.e., it defines a virtual machine. The BLACKBOARD-instructor is an architecture in the second, looser sense since it only specifies that certain software processes correspond to components of an intelligent tutoring system, defines the relationship of these components, and specifies a protocol for communication. In addition to modules for the student model, planner, and domain expert there was another module for a learning element to improve planning behavior. Both the SIIP and BLACKBOARD-instructor *architectures* were implemented but a *planner* was never demonstrated for a realistic and nontrivial application that used these architectures, nor was any learning element ever implemented for those planners that were built in the SIIP architecture. Further discussion of these architectures and a detailed comparison of BB-IP-2 and SIIP is provided in Appendix E.

The first version of the Blackboard Instructional Planner discarded both the SIIP and BLACKBOARD-instructor architectures. Instead an existing architecture (the BB1 Version 2 blackboard architecture [Hayes-Roth 84]) was adopted. The focus of the research shifted from generic ITS/planner architectures to the implementation of a dynamic instructional planner for a realistic domain.

The first such planner implemented was BB-IP-1. It was not based on SIIP but instead on an unrelated blackboard application called PROTEAN [Hayes-Roth 87b]. PROTEAN infers the three-dimensional structure of protein molecules based on constraints including measurements obtained from nuclear magnetic resonance (NMR). It relies on a high-level language of problem-solving operators. These operators are appropriate for solving similar constrained arrangement problems by a method of incrementally assembling solutions. The high-level language was called ACCORD [Hayes-Roth, et al 87]. Hayes-Roth calls such a high-level language, appropriate to a class of problems, a *language*

framework. Extensions to BB1 called BB* simplify the construction of new language frameworks.

The first Blackboard Instructional Planner was built by defining a new language framework called TUTOR. Rather than solving a class of arrangment problems it solves a class of instructional problems. It is intended for tutors that teach the troubleshooting of complex hydraulic-electronic-mechanical devices by first imparting a mental model of the device and its operation using a STEAMER device simulation.

A planner was built using these operators and scenarios were implemented that illustrated lesson content customization, question handling, request handling, context-sensitive tutorial strategies (expository and case-method), and the dynamic adjustment of tutorial strategies. But the planner had the following shortcomings that led to the implementation of BB-IP-2:

- *Instructional actions were only simulated* - so it was difficult to determine if assumptions about actions were realistic.
- *The implementation was slow* - because of time required to interpret the language framework and the complex control structures used.
- *Other tutor components were overly simple* - so it was difficult to draw any conclusions about the relationship of planner to other tutor components that would interact with it.
- *Planning was very restricted* - since only local incremental planning could be performed. The tutor would at most plan out the next few topics and the next few steps in presenting the current topic.

A complicated multi-step instructional plan with multiple lessons and estimated lesson lengths was far beyond the capability of the first Blackboard Instructional Planner. Plan patching and plan critics were also not implemented.

BB-IP-2 has a much more sophisticated plan representation than BB-IP-1 and implements all instructional actions. Other differences include a higher-level of granularity for knowledge source actions, an emphasis on lesson planning rather than discourse planning, and a more sophisticated approach to plan generation and replanning that includes the planning of lesson sequences. Unlike BB-IP-1, there are no knowledge sources to directly carry out didactic actions. Instead, BB-IP-2 knowledge sources only perform planning and plan execution actions and instructional procedures are used to carry out all didactic actions. As a result of the design changes, BB-IP-2 runs much faster than BB-IP-1. A detailed discussion of differences between these two planners is presented in Appendix F.

7.2. Related Research in Planning, Blackboards, and Other Areas

The planning, blackboard, and simulation-based training research most related to BB-IP and the Lower Hoist Tutor is presented here. Some of the research presented, such as IMTS, has influenced the design of the non-planning components of the Lower Hoist Tutor. Other work is more related to just the planner. For example, independent research in applying dynamic planning techniques coupled with an explicit plan representation to non-ITS knowledge communication follows somewhat parallel lines to BB-IP. Johanna Moore's [Moore 89] work on explanation generation exemplifies this area of research. Recent work that extends classical planning (i.e., SIPE [Wilkins 88]) will be reviewed to provide a useful perspective on BB-IP's approach to dynamic replanning. Finally, those blackboard applications that were important influences in developing BB-IP will be discussed.

IMTS provides a set of tools for rapidly constructing intelligent tutoring systems for simulation-based training. It is discussed here since its primary influence is on the Lower Hoist Tutor's interface and student model and not on its planning component (BB-IP). Its influence on the student model has been discussed in Section 3. The IMTS interface has also influenced the way that the STEAMER [Hollan 84] lower hoist simulation has been adapted for use as a student interface. Although the domains are different, the IMTS Helicopter Bladefold simulation discussed in [Towne 89] and the lower hoist device simulation are similar in the following respects:

1. *Use of icons* - graphic images depict device parts and show what state they are in (e.g., a piston icon might be drawn as extended or retracted).
2. *Propagated effects* - part state changes propagate to model similar changes in the device modelled. Display updates mirror changes that occur in the underlying device model.
3. *Simulated troubleshooting tests* - the student can simulate the use of test equipment such as logic probes or pressure gauges.
4. *Simulated part replacement* - the student can simulate the replacement of parts to correct a device fault.

The first two similarities are due to a common heritage (STEAMER) while the last two are borrowed from IMTS. In contrast, here are some key differences between the systems:

1. *Color* - the lower hoist device simulation uses color to show high and low pressure in pipes. IMTS does not yet use color.
2. *Platforms* - IMTS has been developed for Xerox D-machines and is currently being ported to other platforms. The lower hoist

simulation runs on the Symbolics and TI-Explorer family of LISP machines.

3. *Authoring tools* - IMTS provides a general set of authoring tools for constructing device simulations while the Lower Hoist device simulation was built using only the original STEAMER graphics tools.

4. *Domain expertise* - IMTS provides general domain expertise for inferring the propagation of part state changes, and for determining optimal troubleshooting actions. IMTS can also simulate faulted device operation; the lower hoist device simulation cannot. No general troubleshooting expert is yet available for the lower hoist.[7]

Although IMTS can select optimal troubleshooting actions it cannot participate in an dialog to explain why its choice is best and why another choice is not. Explanation generation is another research area where dynamic planning techniques, explicit plan representations, and the use of blackboard architectures are being explored. Moore [Moore 89] advocates the use of an explicit plan representation and dynamic planning for generating explanations for expert systems. Dynamic planning is required to further elborate an initial explanation, to clear up misunderstandings, and to answer user questions about the explanation. As in BB-IP, an explicit representation of intended goals is used in replanning. Other recent work in natural language generation [Nirenburg 89] takes advantage of the blackboard architecture's ability to support opportunistic planning (discussed below). Although BB-IP relies on a blackboard architecture it is not currently an opportunistic planner.

BB-IP is built using the BB1 blackboard architecture [Hayes-Roth 84]. BB1 in turn derives from research on a planner called OPM [Hayes-Roth 85], which simulates human errand-planning. The planner pieces together a plan to handle a set of errands in a fictional town. There is not enough time to perform all the errands. The errands have different priorities and different locations in the town. The planner selects an overall strategy (such as perform tasks in the southwest corner of town first) but can deviate from a high-level strategy to take advantage of unforeseen opportunities. E.g., if the bank is nearby the post office both errands may be performed even if this causes a deviation from the original strategy. This kind of ability to take advantage of unforeseen opportunities that

---

[7]Work is underway to provide such expertise using an approach based on truth maintenance systems. Basically, contradictions between the expected device state and actual device state lead to the discovery of those parts or sets of parts that may be faulted and which can resolve the apparent contradictions.

arise in planning is what is meant by *opportunistic planning*. Opportunistic planning should not be confused with *opportunistic tutoring*, which takes advantage of pedagogical opportunities that arise in the course of tutoring. It is not clear how to usefully apply opportunistic planning to instructional planning, and BB-IP does not do so at this time. Later research on BB1 led to the development of language frameworks and the BB* environment that support these high-level problem-solving languages. This research in turn influenced the first version of the Blackboard Instructional Planner (BB-IP-1), as discussed earlier.

Although BB-IP-2's approach to plan generation does not resemble OPM's approach, its approach to dynamic replanning does resemble that of the SIPE planner [Wilkins 88]. SIPE extends classical planning techniques based on STRIPS [Fikes and Nilsson 71] and NOAH [Sacerdoti 77] by incorporating:

- *Heuristics for efficiency* - these allow SIPE to generate plans rapidly enough (in a matter of seconds) to be used in practical applications.
- *A deductive causal theory* - this simplifies operator representations by allowing context-dependent effects of actions to be deduced.
- *Reasoning about resources* - this allows reasoning about the cost of actions (e.g., fuel or time).
- *Constraint posting and propagation* - this allows a least-commitment strategy to planning that cuts down on unnecessary backtracking. Objects that are to be used in actions may be only partially specified. Later constraints may narrow the choice of acceptable objects to only one.
- *Plan critics* - these allow incorrect plans to be detected and (if possible) corrected. Backtracking occurs if no acceptable plan can be found.
- *Replanning* - SIPE provides domain-independent strategies for repairing or improving a plan.

BB-IP's approach to replanning differs from SIPE's primarily by its incorporation of a diagnosis phase to determine the cause of plan failure. In contrast, SIPE assumes that a complete description of all unexpectedly violated predicates is available; these predicates are entered by the user. However, SIPE's classification of plan execution problems and its terminology is useful in understanding BB-IP's diagnosis of similar errors and the plan repair methods it uses.

Here are some of SIPE's classification of plan execution errors, its replanning responses, and the BB-IP analogs:

- *Purpose not achieved* - Some action fails to achieve its intended effects. SIPE calls its planner to produce a new plan for that part of the plan that failed. BB-IP analog: when assessment immediately following presentation indicates the student did not acquire the skill just covered, BB-IP replans to cover it again. A different procedure is used if possible.

- *Previous phantom untrue* - A *phantom* is a goal that is believed true at some point in the plan. If such a goal is no longer true and should be then SIPE attempts to achieve the same goal using different variable bindings for objects in actions. If that fails it will attempt to achieve the original goal by calling its planner. BB-IP analog: when the student asks a question indicating that he has forgotten prerequisite material, the prerequisite material is reviewed. The prerequisite material is similar to SIPE's phantom goals if the tutor assumed that the student knew it based on the pre-instruction questionnaire.

- *Future goal already satisfied* - Some future goal no longer needs to be achieved since it is already true. The part of the plan to achieve the goal is removed. BB-IP analog: when the student performs discovery activities and covers material that BB-IP intended to cover, then those activities and procedures to cover the redundant material are marked as deleted.

Other kinds of SIPE plan execution errors discussed in [Wilkins 85] do not have BB-IP analogs and so are not discussed here.

Although SIPE provides many planning capabilities required by dynamic instructional planners, extensions would be required to implement planners such as BB-IP. SIPE's efficiency, hierarchical plan representation, and its ability to reason about resources, to replan, and to plan incrementally all provide some support for dynamic planning. However the following do not yet appear possible in the planner architecture described in [Wilkins 88]:

- *Plan critics that improve plan utility* - SIPE has no notion of plan utility. Plans are either correct or not. Its plan critics detect violations of resource constraints and unacceptable orderings of actions. In contrast, the plan critics in BB-IP improve the utility (i.e., discourse coherence) of an instructional plan.

- *Planning actions to monitor plan progress or diagnose plan failure* - SIPE assumes that the user types in a complete description of all violated predicates to indicate planning failures. BB-IP cannot make such assumptions. It must insert assessment actions into its plan to monitor its progress and it must decide how to diagnose plan failures.

Of course SIPE could be *extended* to overcome these limitations, although it is not clear how difficult this would be. Possible extensions to BB-IP to perform opportunistic planning and sophisticated meta-level reasoning about replanning might also be much more difficult to replicate in SIPE than to implement in the current blackboard architecture.

8. Conclusions and Future Directions

This section summarizes the conclusions of this research, discusses some unanswered questions, and considers future research to address these questions. Observations on the instructional planning task and instructional planners are presented in Section 8.1. Section 8.2 discusses limitations of the Blackboard Instructional Planner and the Lower Hoist Tutor, and future directions for research. The final section, Section 8.3, summarizes the research contributions of this work.

8.1. Nature of Dynamic Instructional Planning

Dynamic instructional planning is an instance of a larger class of dynamic planning problems characterized by an environment that is

- *Incomplete* - since the tutor has limited access to the student's mental state,
- *Uncertain* - since this access is indirect and the tutor's inferences may be wrong,
- *Dynamic* - since the student's knowledge changes during and between tutorial sessions,
- *Multi-agent* - since the tutor and student cooperate to facilitate the student's learning,

and where

- *Results of actions are uncertain* - since the tutor cannot predict with certainty the results of its actions.

Other examples of problems sharing these characteristics include battle management, air traffic control, natural language discourse, crisis management, and legal argumentation. In some ways instructional planning is not as difficult since multi-agent planning and real-time stress are less important issues. There are only two agents involved (more in a collaborative learning environment, or if the classroom instructor interacts with the instructional planner), and real-time stress is only moderate. Furthermore, the student and tutor are in a cooperative rather than adversarial context.

Instructional planning is also simpler than other planning problems due to the limited reasoning about parallelism. SIPE can generate complicated nonlinear plans allowing parallel execution by multiple agents. Such plans could be used to control several robots at once in an assembly task. This kind of parallelism does not need to be dealt with in the one-on-one tutoring context assumed in this research. If multiple interacting tutors were used to teach students, then such planning would be required.

Reasoning about resources is simpler because typically only time needs to be considered. In most instructional planners even time is not considered and then planning can be very simple. Top-down plan expansion can be used as in IDE-INTERPRETER [Russell 88a] or simple goal-driven planning as in Peachey and McCalla's planner [Peachey 86]. However, such methods by themselves would be inadequate if nonlinear plans and resource conflicts were allowed. Additional methods, such as SIPE's use of plan critics and backtracking, would be required then. BB-IP *does* reason about a single resource - time - but its approach is quite simple since it does not handle nonlinear plans nor does it provide a general mechanism for handling resource conflicts.

Most plan representations used in instructional planning are hierarchical. Some only represent lesson content but not means of delivery (e.g., IDE-INTERPRETER). Others, such as BB-IP, represent both content and delivery. Some represent plan contigencies (e.g., the non-hierarchical plan representation used in [Peachey 86]) and others do not (e.g., BB-IP). The tutor's intentions can either be explicitly represented or not. When these intentions *are* represented, replanning is facilitated. The tutor can reason about what should have happened, why it should have happened, and what to do about it. Alternatively, special rules to handle particular kinds of errors can be provided (e.g., the error recovery language of SIPE [Wilkins 35]).

The key point is that fairly simple top-down expansion planners may be sufficient and easily implemented for many ITS systems. Complexities are introduced when time is important, when student interrupts can overturn plans, and where fine-grained diagnosis and replanning is required for remediation. Then more complicated planners are required, and the blackboard architecture is useful for implementing such planners.

The blackboard architecture is a software tool that simplifies the construction of dynamic instructional planners by providing a hierarchically-structured global database, independent knowledge sources, and agenda control. The database supports the instructional plan representation. The knowledge sources can act as plan refinement operators, plan critics, monitoring demons, plan diagnosis experts, and plan editors. The agenda control mechanism supports

meta-level reasoning when there are conflicting goals and several competing actions to choose among. Of course, because of Turing compatibility, the blackboard architecture is not required. BB-IP could have been implemented from scratch, by extending SIPE, or in MRS. But these implementations would have been much more difficult.

The SIIP and BLACKBOARD-instructor architectures proved too complicated and abstract to facilitate implementation of dynamic instructional planners. The multiprocessing of the BLACKBOARD-instructor was especially cumbersome. Experience with these architectures suggests that research constructing general-purpose ITS architectures is only useful if an application can be built that demonstrates the utility of the architectures and provides specific guidelines on how to use the architectures.

8.2. Limitations and Future Directions

There are still many unresolved issues that a planner such as BB-IP raises. How much planning should be performed before instruction begins? How often should assessment be performed to monitor the student's progress? What is the best approach to diagnosing plan failure when the student is not learning as expected? The planner provides a framework to explore these issues.

The SIIP research and BB-IP research does not address pedagogical issues regarding the best means of initial student assessment, plan customization, diagnosing plan failures, or providing remedial instruction. The SIIP planner, BB-IP-1, and BB-IP-2 each make arbitrary decisions in these areas that have surface plausibility but which are not based on any pedagogical theory. These planners could be modified to test different theories by altering their knowledge sources. Since this has not yet been done in any nontrivial way[8] a future direction of research would be to attempt to operationalize and evaluate alternative pedagogical theories to guide plan generation and replanning in a dynamic instructional planner.

Limitations in the Blackboard Instructional Planner (version 2) suggest other avenues for future research. These limitations are:

- *Interruptibility* - interrupts are only allowed between procedure steps. Ideally, the student should be able to interrupt at *any* time.

---

[8]The Geometry Tutor described in Appendix E attempted to embody Bruner's theory of concept attainment as described in [Joyce 86]. However, it was unclear how to operationalize several parts of the theory and the tutoring task was overly simple.

- *Meta-level reasoning* - is largely absent. The planner will use the same replanning method repeatedly even if it fails the first time.
- *Plan diagnosis* - is quite primitive. Better approaches to diagnosing plan failure and planning remedial instruction are needed. Planning for diagnosis should be addressed.
- *Reasoning about time* - is unrealistically simple at present. The application of scheduling algorithms could be investigated here.
- *Robustness* - is inadequate for classroom use. When not running scenarios, the planner tends to either not replan when it should or to replan when it should not.

The lack of robustness is caused by the difficulty in determining the effect that assessments should have on the student model and in determining when replanning should occur. Currently replanning occurs when the certainty factors associated with a student model skill either:

- *Exceed a threshold* - indicating that the tutor now believes the student has acquired the skill.
- *Fall below a threshold* - indicating that the tutor believes the student does not have the skill.
- *Fall when they should rise* - indicating that the tutor believes that the student is not making progress towards acquiring the skill.

It is not clear what these thresholds should be or how to update the student model based on assessments so that the tutor will draw the proper conclusions about the student's learning.

Future research can provide a more principled approach to student modeling in planner-based tutoring systems, integrating a means of updating the student model to incorporate new assessment information with a means of deciding when replanning is appropriate. Such research is necessary before the current approach can scale up to the point that systems can be built for classroom use. The current use of numbers to aggregate possibly conflicting assessment information accumulated over a tutorial session is inadequate. A better approach would be to represent the tutor's beliefs about the student's beliefs along with the justifications for the tutor's beliefs. These justifications would refer to tests, student self-assessment, student-initiated questions and requests, results of the pre-instruction questionnaire, and student performance on troubleshooting. A sophisticated inference mechanism would be required for deciding how to incorporate new assessments and to decide whether or not replanning is justified. The use of a truth maintenance system (TMS) appears promising for this student modelling application. If a replanning approach failed or if later assessments

contradicted tutor beliefs inferred from cognitive stereotypes, the TMS could be used to isolate the faulty assumptions, revise them, and remove dependent inferences that would otherwise lead to further inappropriate tutorial actions. Student diagnosis could also benefit since uncertainty about possible erroneous assumptions could guide the tutor's questions. Finally, the tutor could use the TMS to explain its replanning actions in terms of its beliefs about the student. These explanations could be used either to assist in the debugging of the planner or to explain to the student why the lesson plan is being changed.

8.3. Research Contributions

This section discusses the publications, software, and academic contributions of this research. First we consider publications.

Macmillan and Sleeman's design for the SIIP planner architecture and BLACKBOARD-instructor ITS architecture is presented in *Computational Intelligence* [MacMillan 87]. Operation of the SIIP planner is also discussed in that paper. The book chapter appearing in [MacMillan 88] is largely the same but includes an expanded discussion characterizing the instructional planning process.

Research on BB-IP-1 is first presented in an FMC technical report [Murray 88c]. An expanded version of this paper appears in the Proceedings of the 4th International Conference on Artificial Intelligence and Education [Murray 89b]. It includes a survey and analysis of alternative approaches to control for intelligent tutoring systems. That paper received the conference best paper award and was later reprinted in ECCAI's journal *AI Communications* [Murray 89c]. A very similar paper with less emphasis on ITS and more on the blackboard implementation appears as a book chapter in [Murray 89a]. A different paper comparing discourse management networks and blackboard architectures appears in *Journal of Machine-mediated Learning* [Murray 88a]. This ONR final report is the first publication describing the second version of the Blackboard Instructional Planner.

Source and object code for the Blackboard Instructional Planner is available from the author. Recipients must first sign an agreement with Stanford's Office of Technology Licensing to obtain permission to use BB1. The BB-IP code will run on any Symbolics LISP machine, including the MacIvory. Code for the lower hoist simulation is FMC proprietary and cannot be distributed without special arrangement. The planner can run without the lower hoist simulation, in which case the graphics commands are simulated and no color monitor is required.

Now we consider academic contributions. The first major contribution of this research is its analysis of architectural and planning issues in ITS. This

analysis is presented both in the papers discussed above and in Section 6 of this report. Alternative approaches to planning for ITS, and trade-offs made between them, are considered in [Murray 89b]. Two particular architectures for intelligent tutoring systems, discourse management networks and blackboard architectures, are analyzed in [Murray 88a]. That paper concludes that blackboard architectures provide greater support for the implementation of dynamic instructional planners, although discourse management networks are sufficient for sophisticated dialog management where only local planning is required.

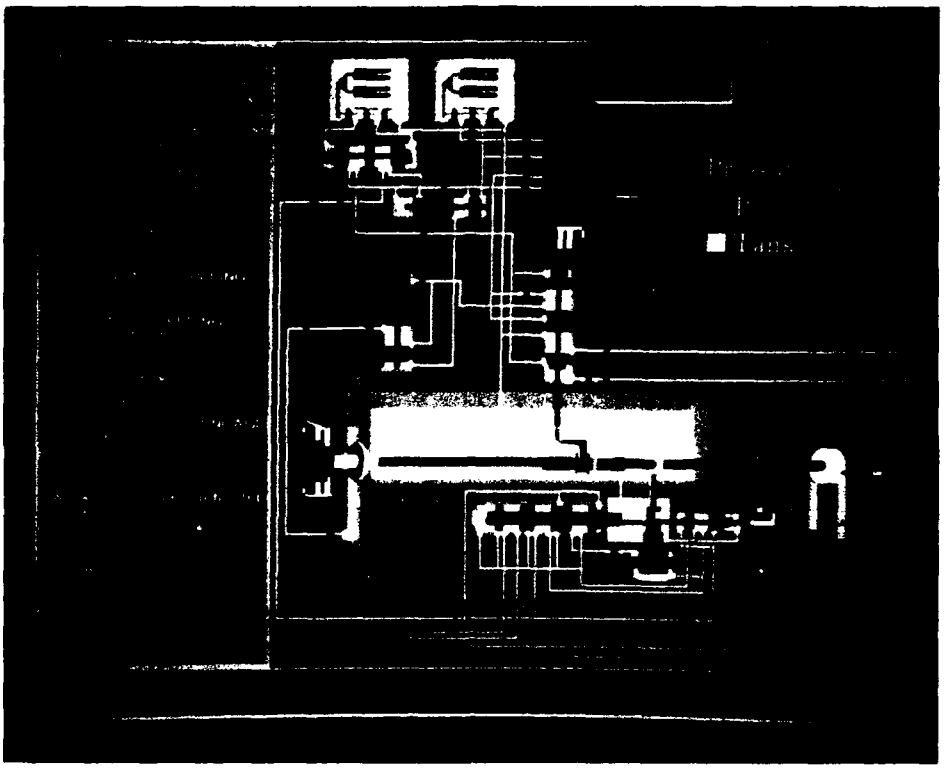
A second research contribution is the implementation of the Lower Hoist Tutor. This prototype tutor shows how a dynamic instructional planner such as BB-IP can be integrated with other ITS components to form a complete system. A realistic naval training application, teaching troubleshooting of the lower hoist assembly of the Mark-45 naval gun mount, has been selected to test the planner. Although the tutor is experimental and not yet ready for classroom use, all instructional actions have been implemented and the tutor can deliver about two hours worth of instruction on the lower hoist.

The major research contribution of this work is the implementation of the Blackboard Instructional Planner. This planner shows precisely how a dynamic instructional planner can be implemented in the blackboard architecture. The Blackboard Instructional Planner generates an instructional plan customized to student background and time constraints, then adaptively replans to support mixed-initiative instruction, and to handle changes to the student model and time remaining.

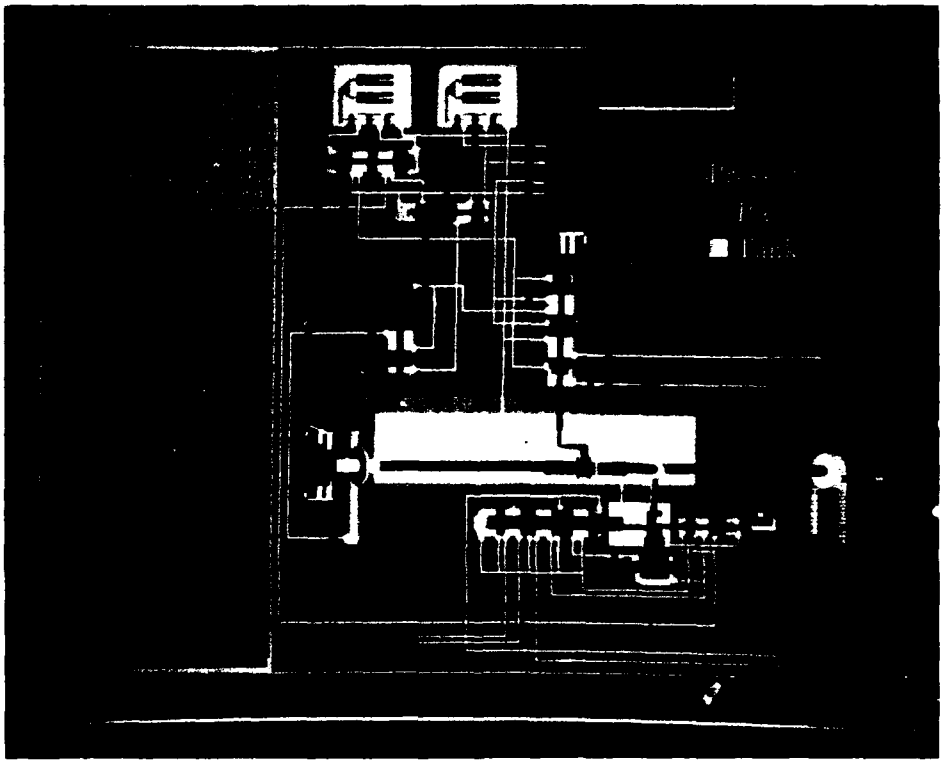
A. Illustrations of Plan Execution and Interruption

The color plates shown in Figure A-1 show the execution of one of the instructional procedures in the first lesson plan of Appendix B. This is the procedure called *Cycle-Overview*. It provides an overview of the six subcycles that occur in the normal operation of the lower hoist. The illustrations show only the beginning part of the procedure up to the explanation of the second subcycle. Note that the student's attention is drawn to the most important changes that occur in each subcycle and that intermediate part state changes are ignored. Figures A-1 (a) - (i) each correspond to a different procedure step.

The color plates shown in Figure A-2 show part of the execution of the procedure *Explain-Subcycles* and its interruption by a student question. This procedure is similar to *Cycle-Overview* but elaborates on *all* the changes that occur in each subcycle, explaining each part state change. The student interrupts with a question about the role of the UVK4 coupling actuating piston (Figure 1-5 shows the lower hoist schematic and where this part is located). The tutor answers the question and reviews related material on part roles since it appears that the student may have forgotten this material. After the review, the tutor presents a transition statement to provide a smooth context shift back to the resumption of the *Explain-Subcycles* procedure, at the point where it was interrupted.

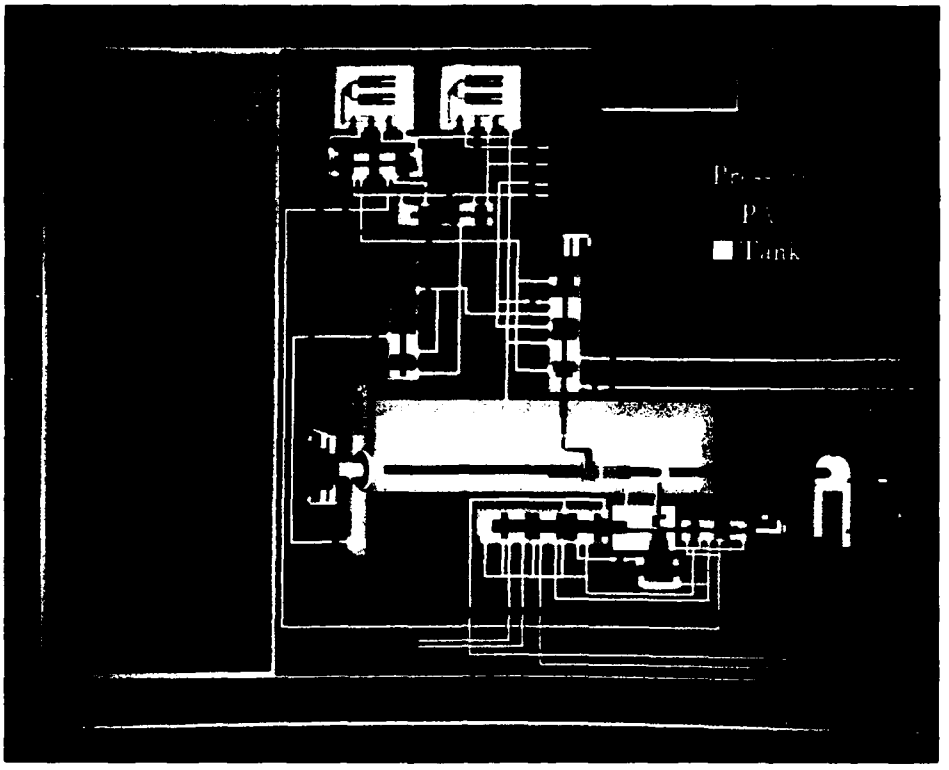


(a)

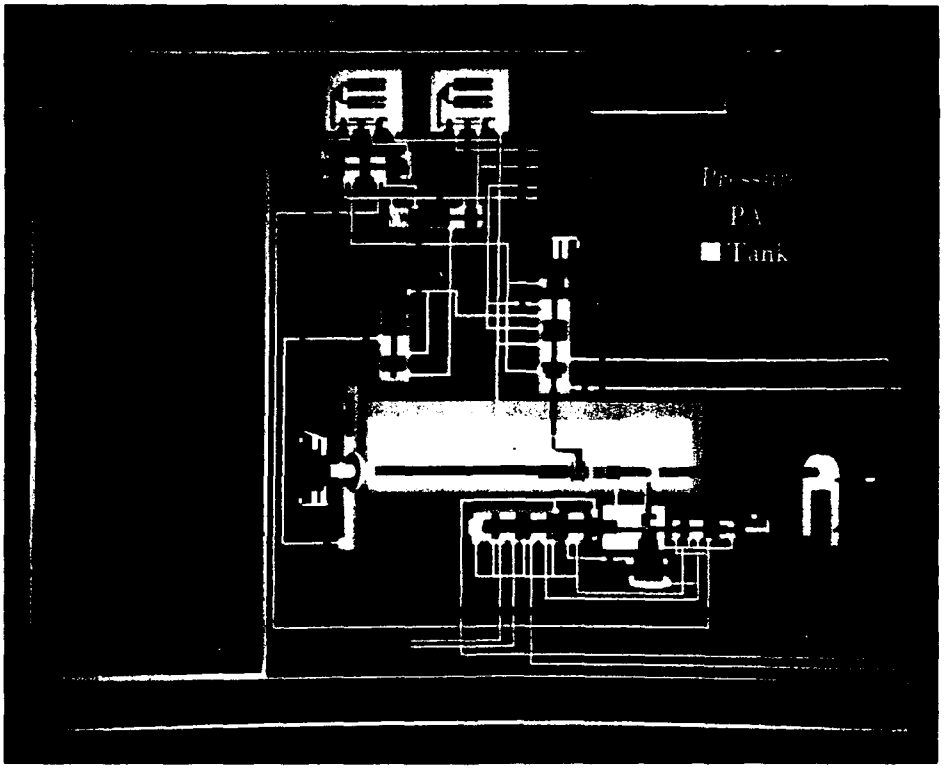


(b)

Figure A-1: Procedure steps in execution of *Cycle-Overview* procedure

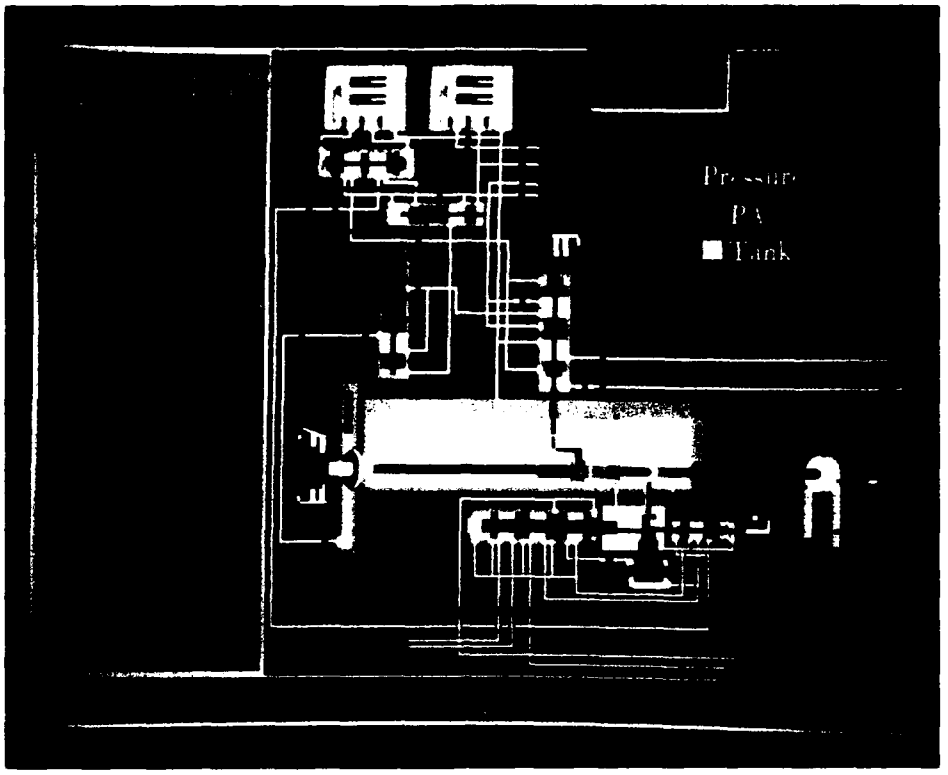


(c)

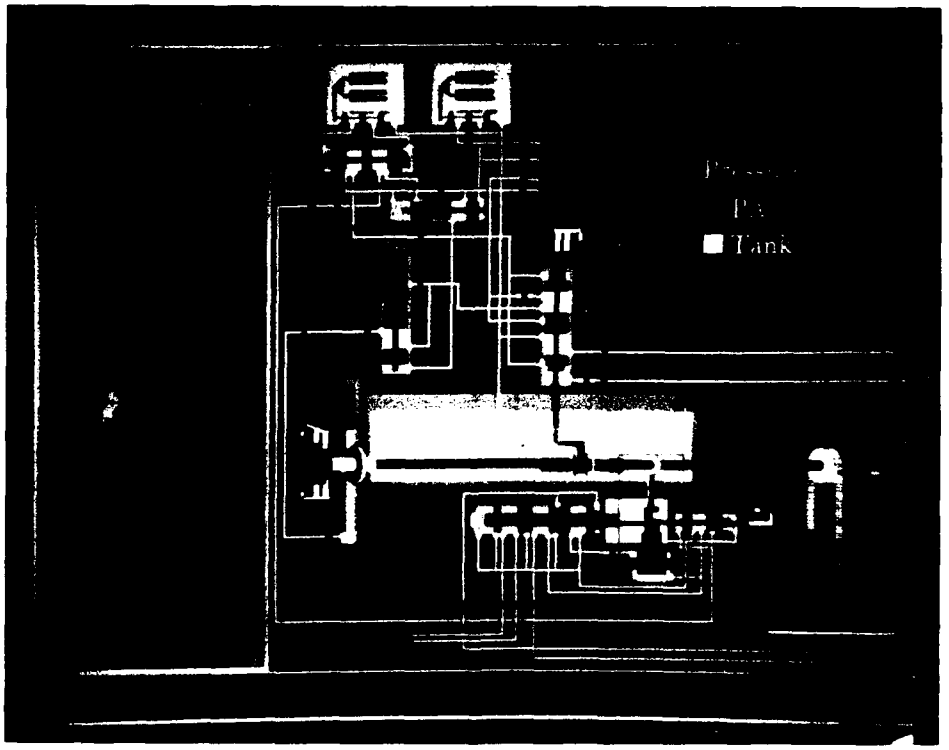


(d)

Figure A-1, continued

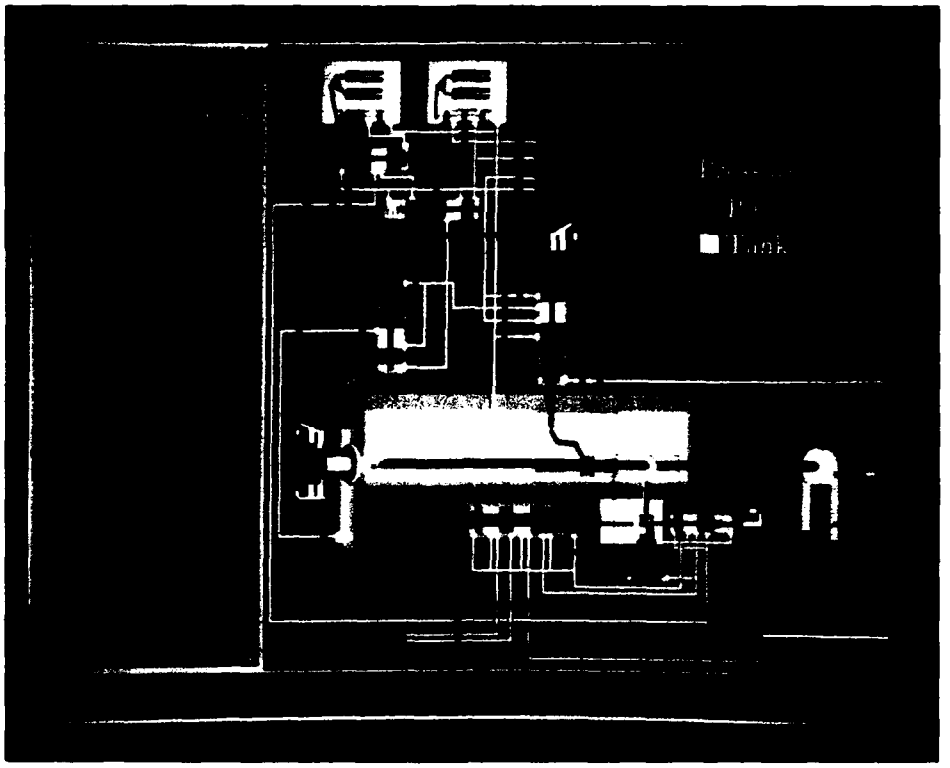


(d)

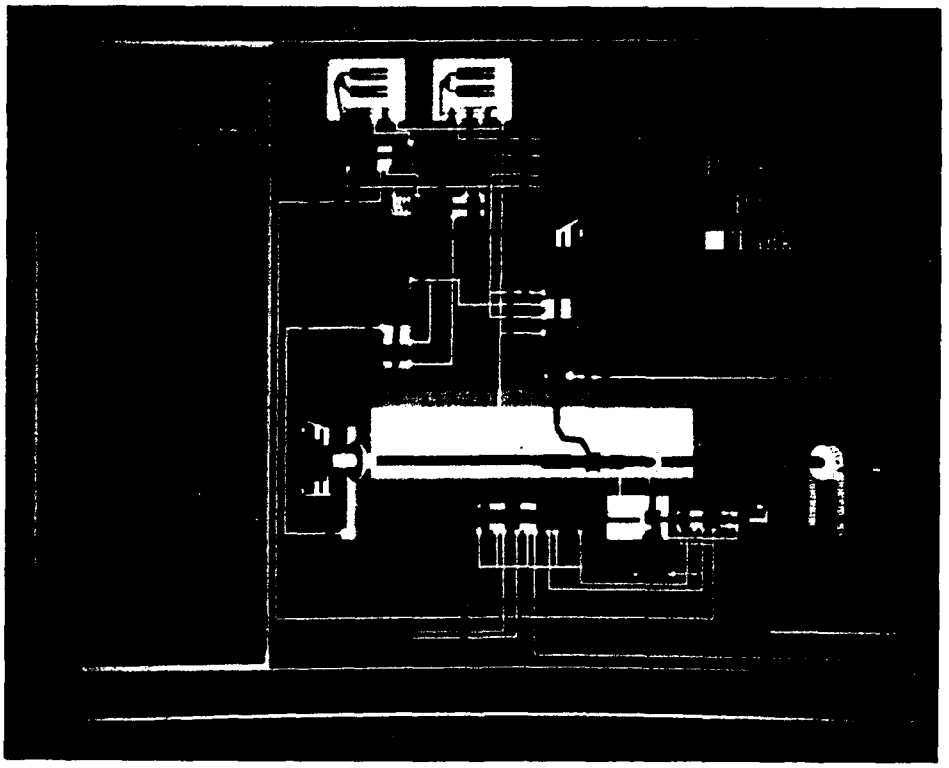


(e)

Figure A-1, continued

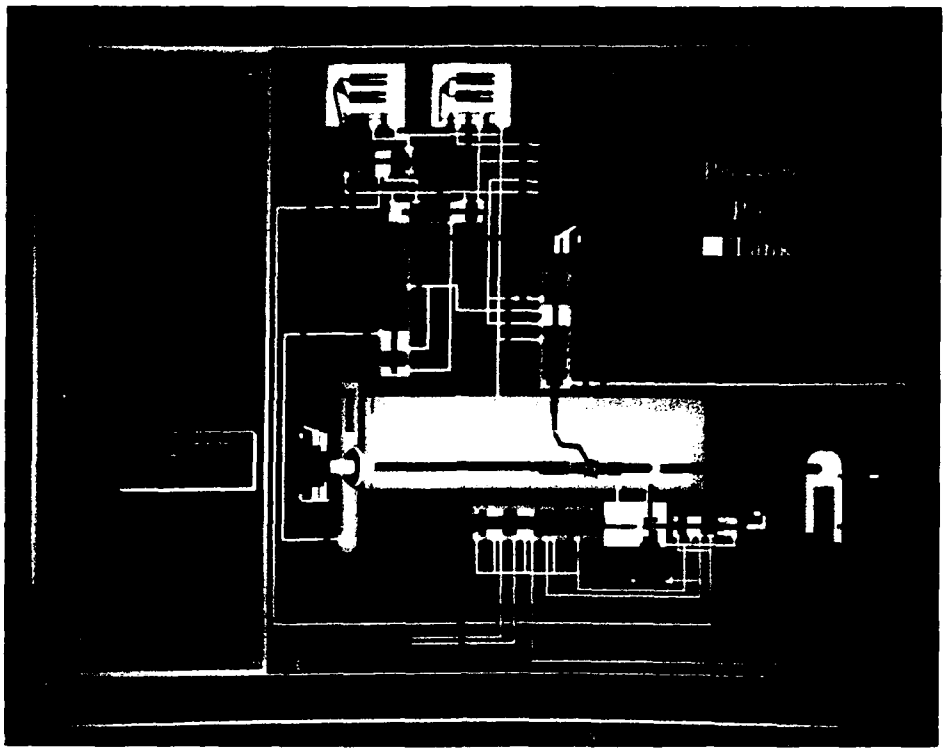


(f)

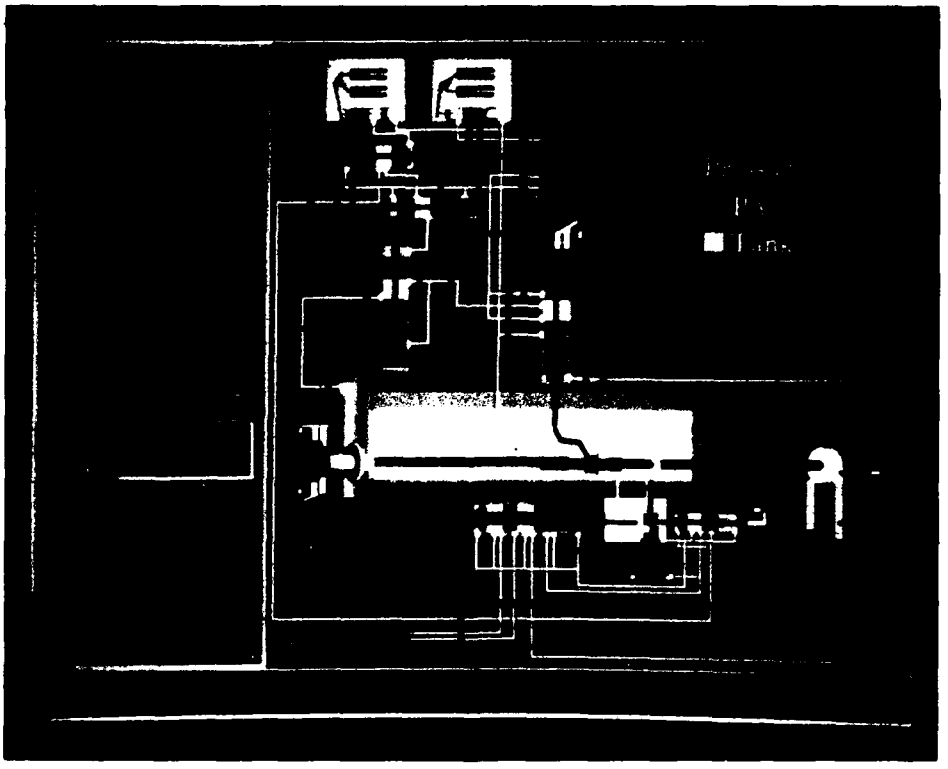


(g)

Figure A-1, continued

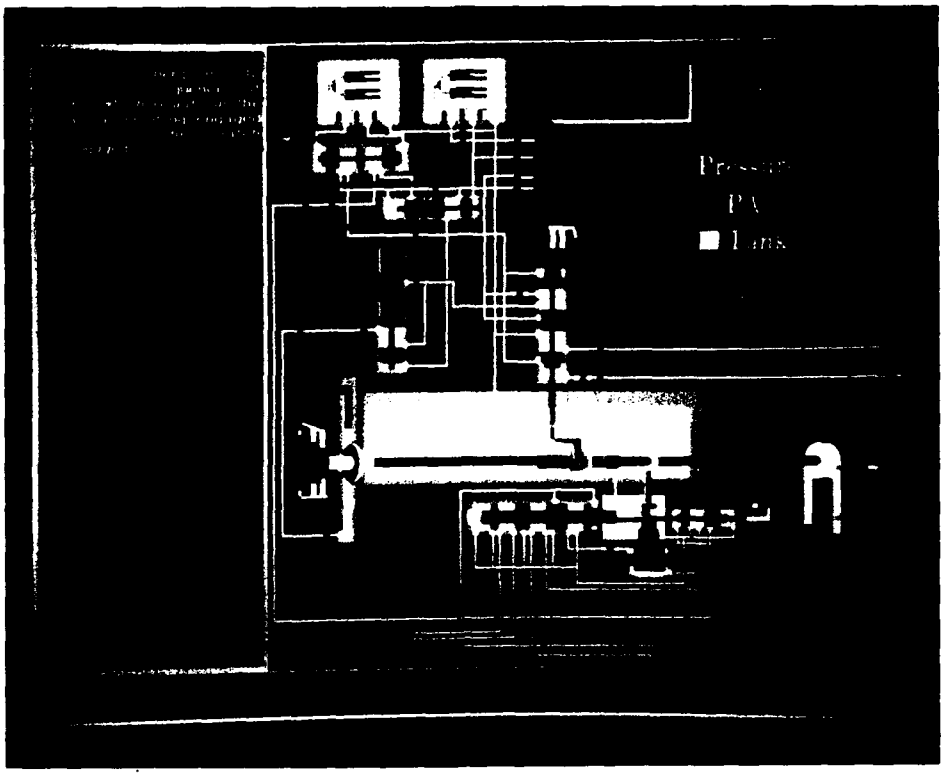


(h)

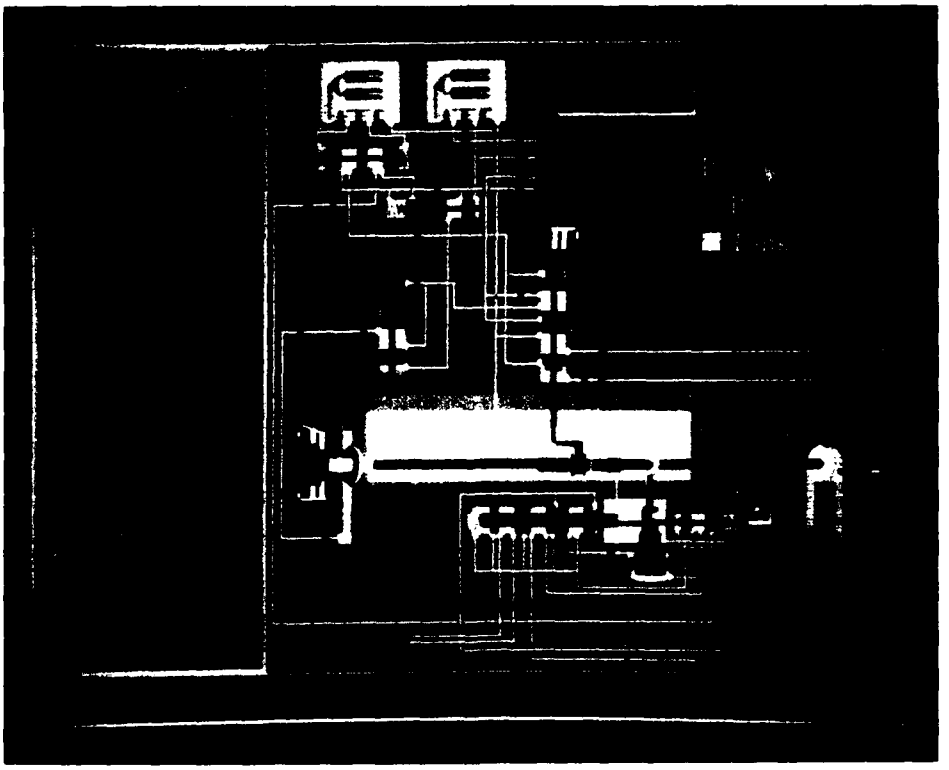


(i)

Figure A-1, continued

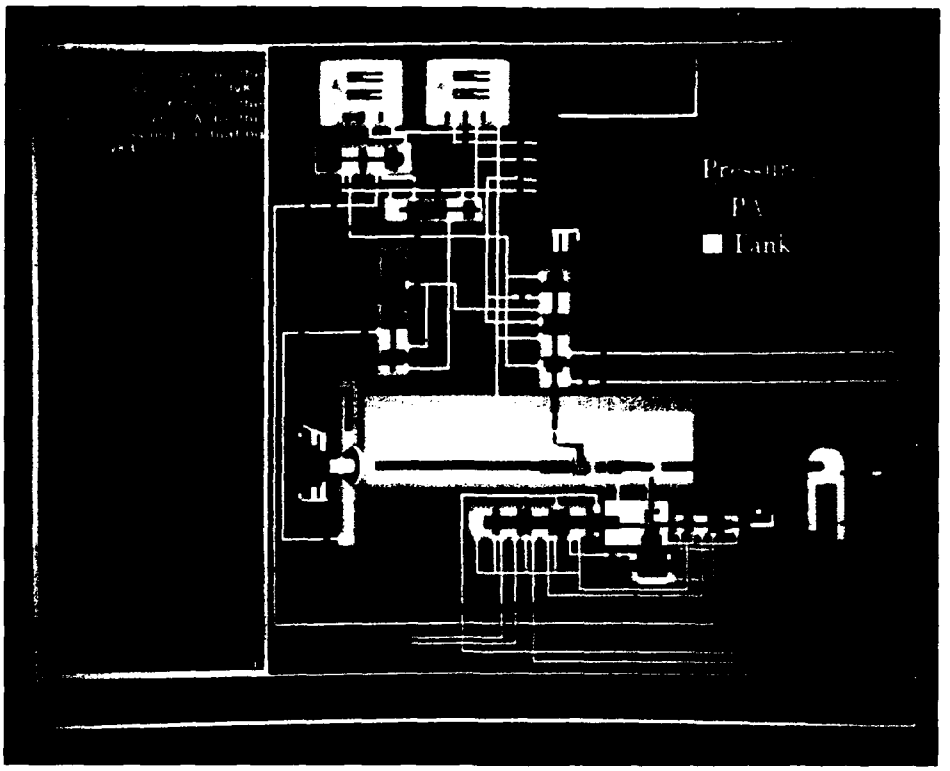


(a)

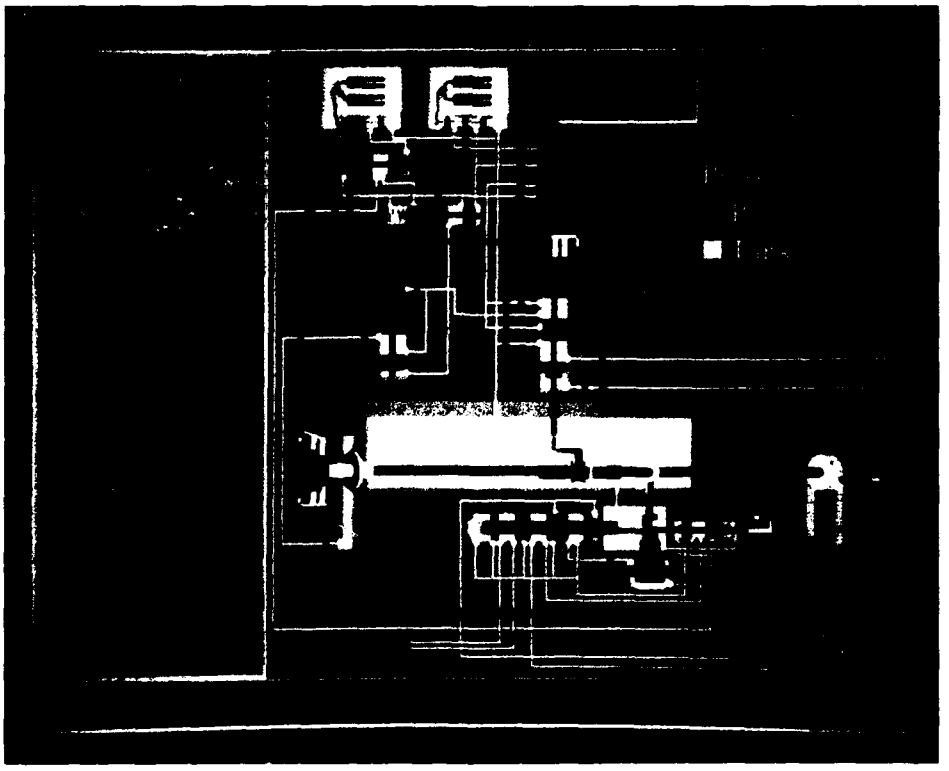


(b)

Figure A-2: An instructional procedure interrupted by a question

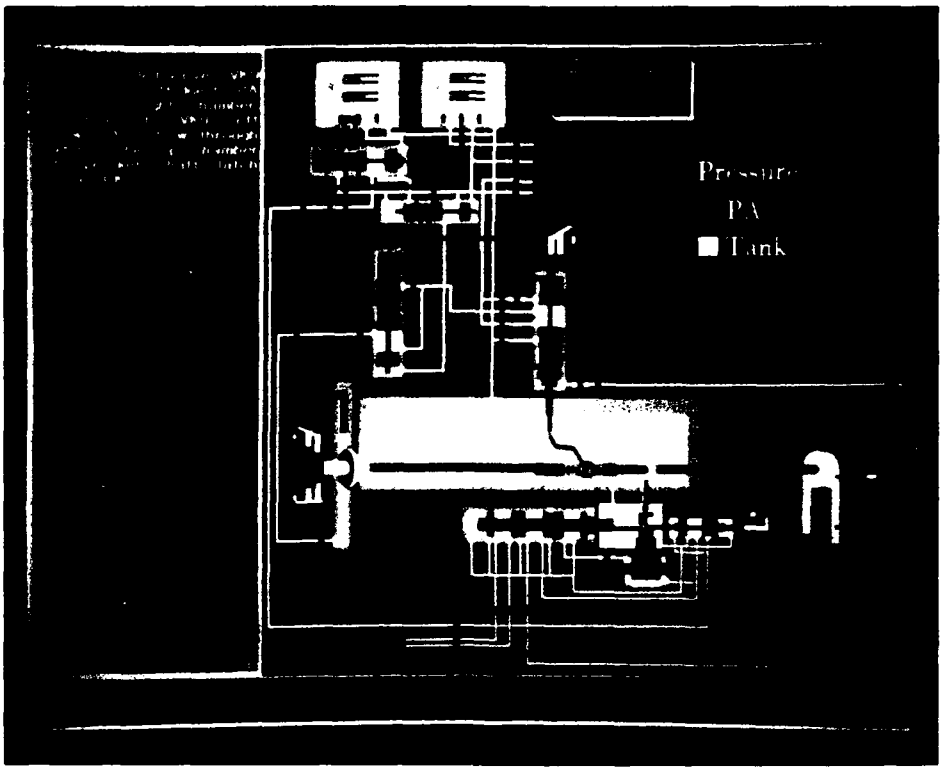


(c)

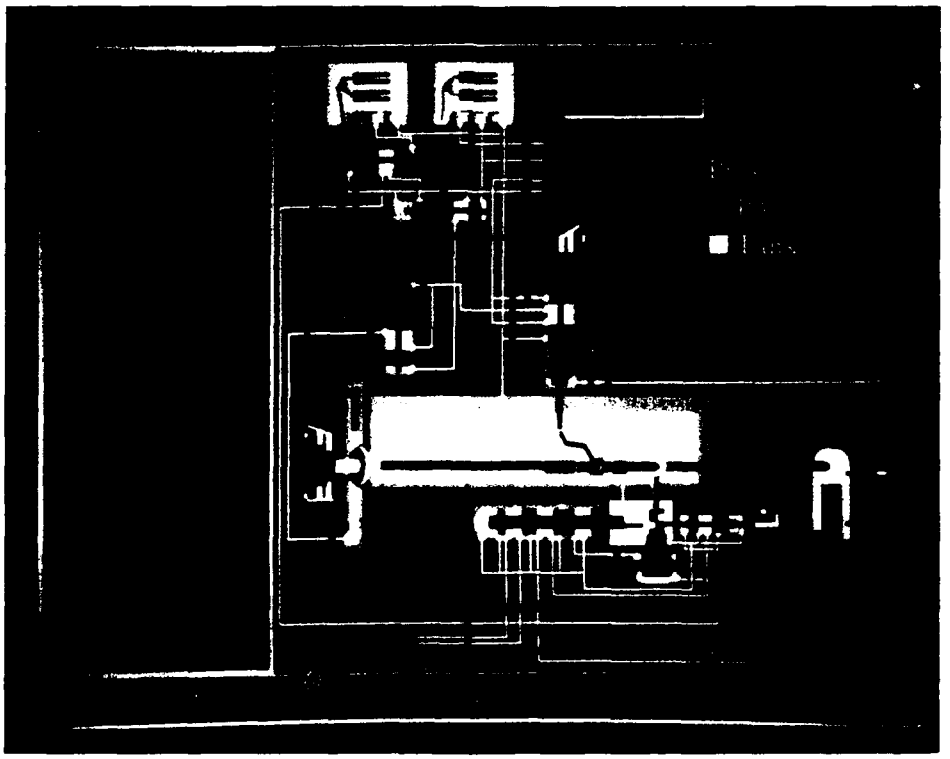


(d)

Figure A-2, continued

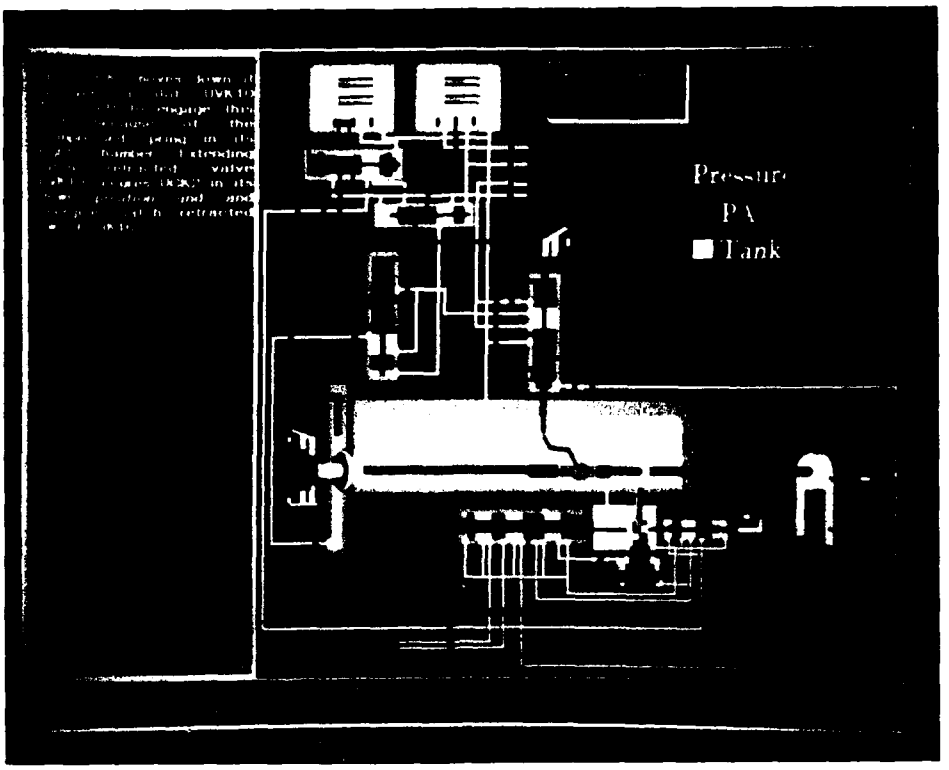


(e)

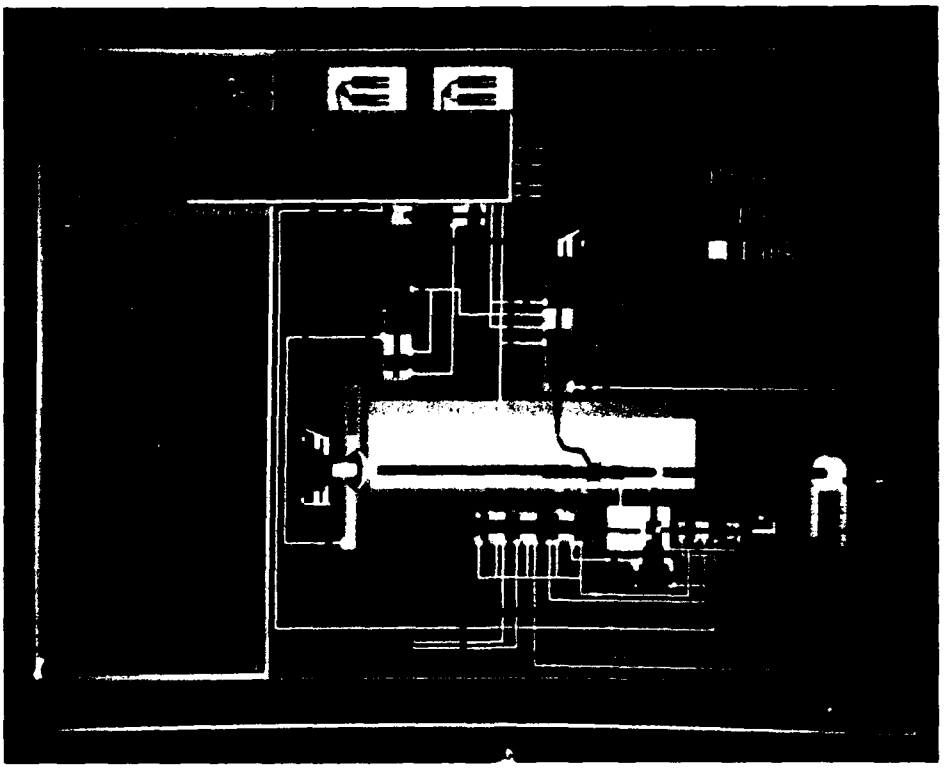


(f)

Figure A-2, continued

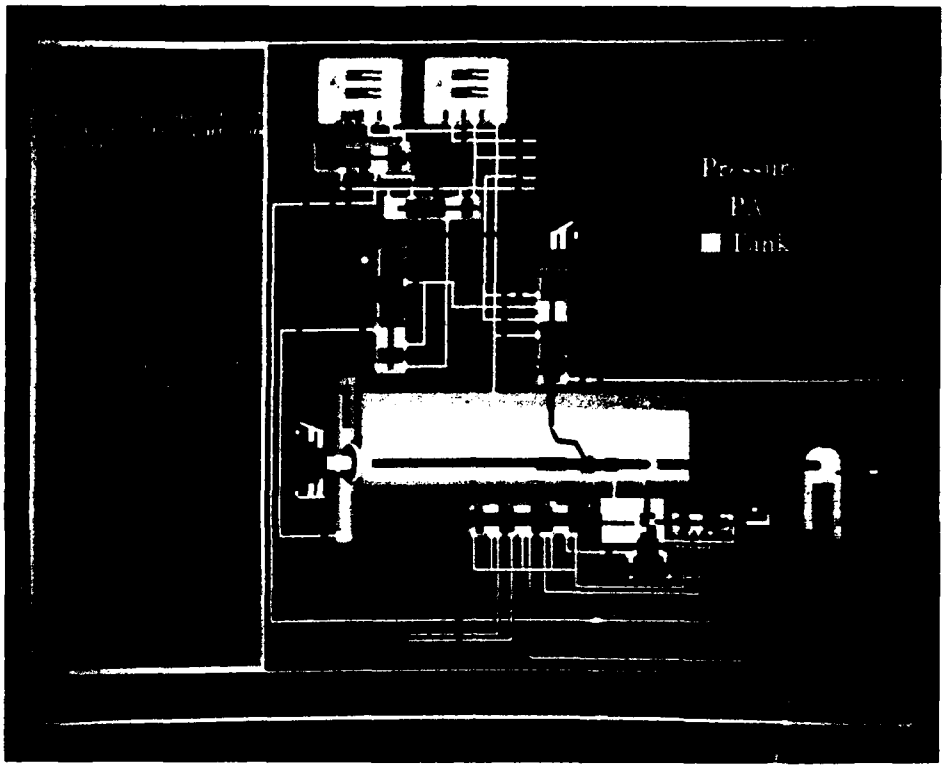


(g)

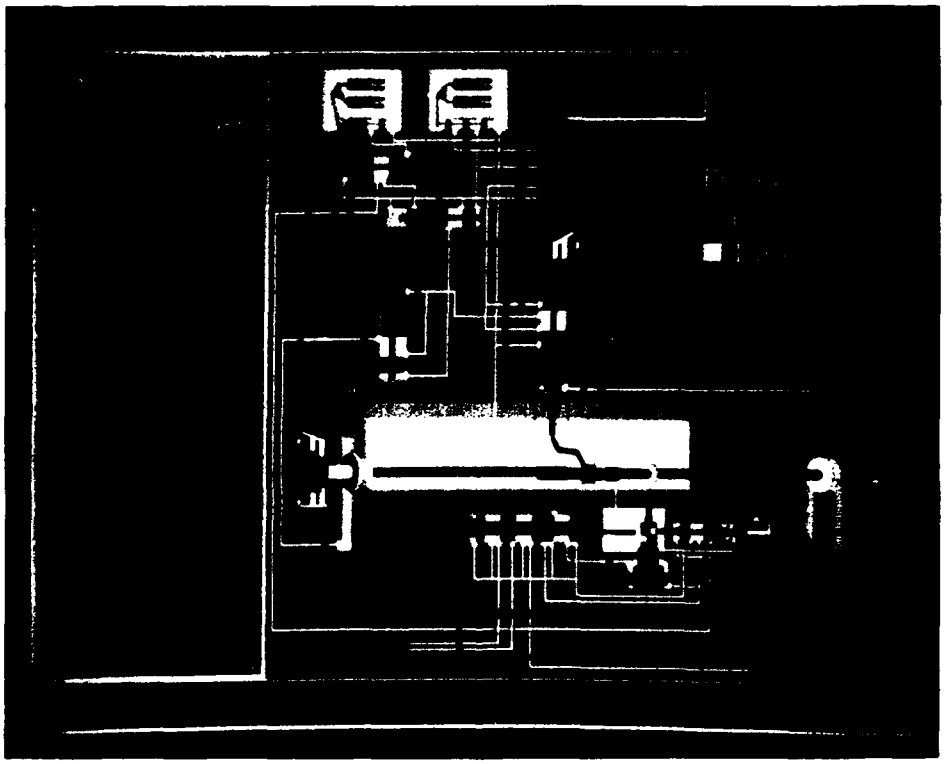


(h)

Figure A-2, continued

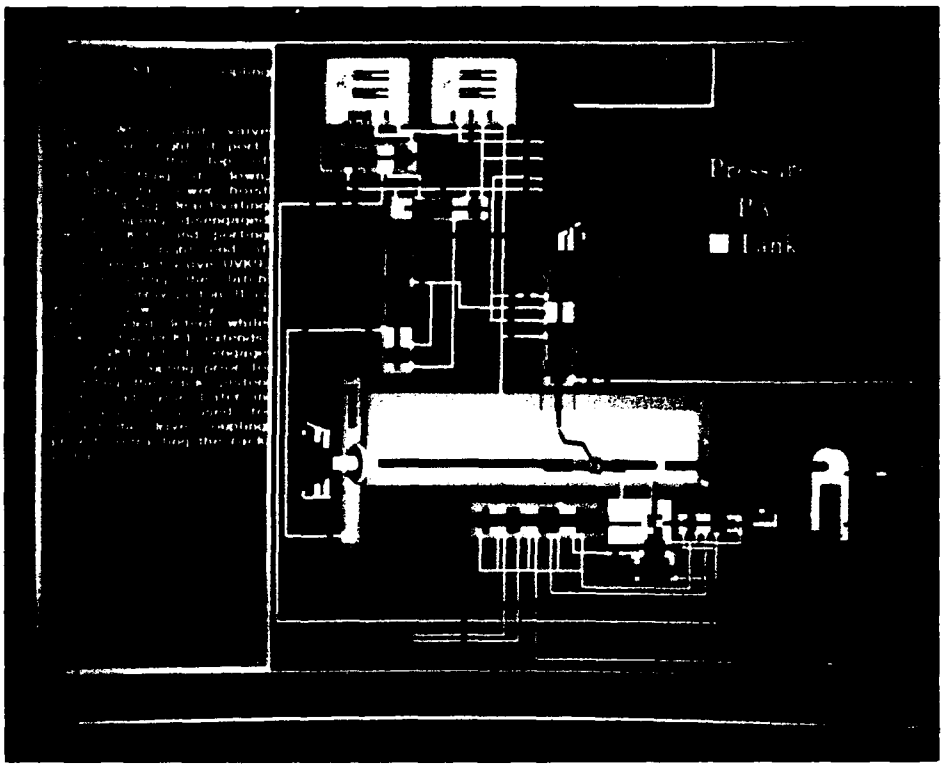


(i)

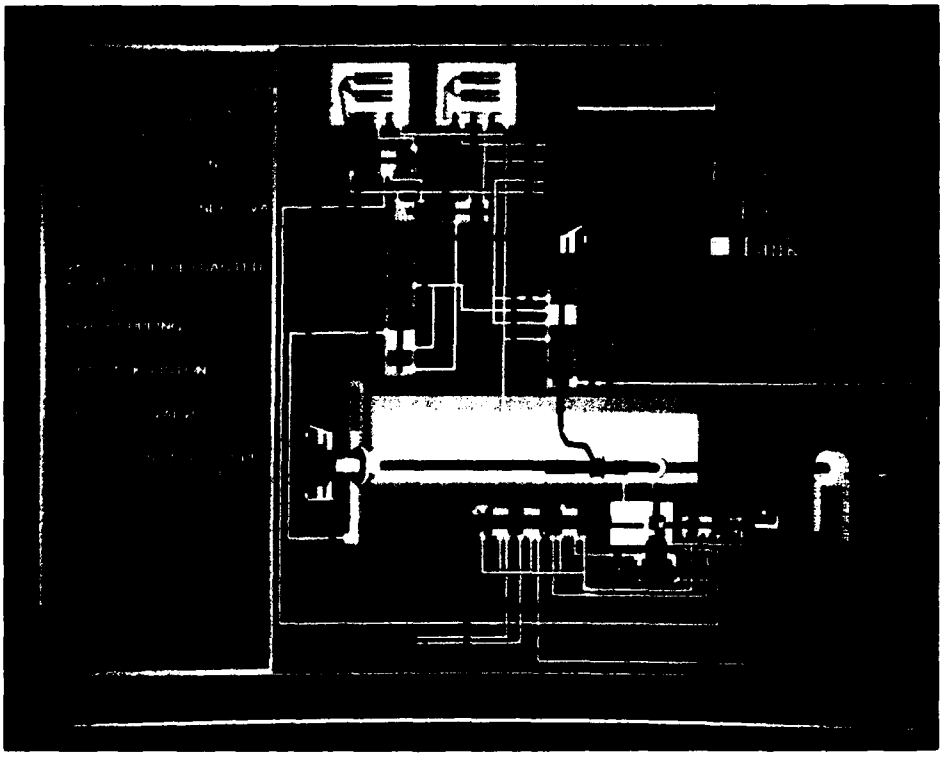


(j)

Figure A-2, continued

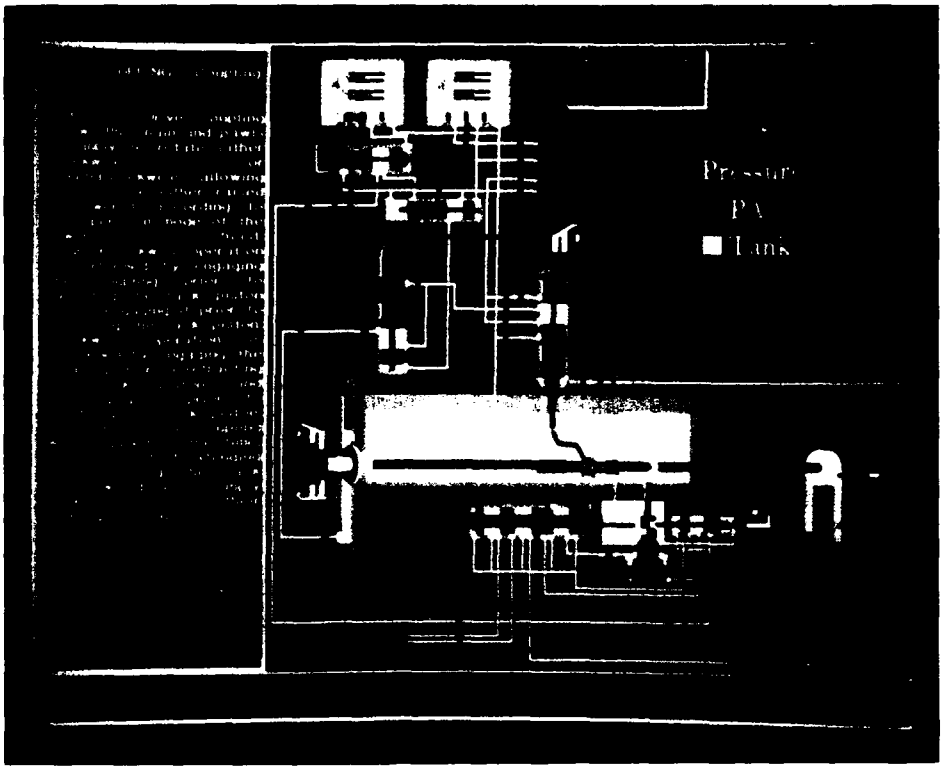


(k)

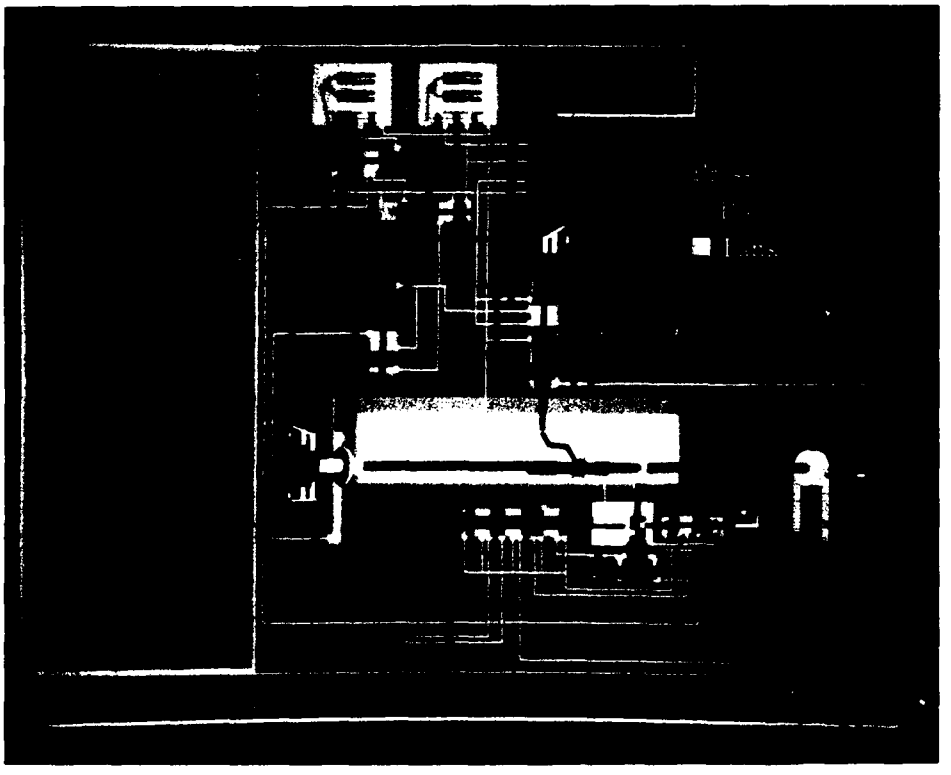


(l)

Figure A-2, continued



(m)



(n)

Figure A-2, continued

B. Two Examples of Customized Instructional Plans

We consider two different student backgrounds and the resulting lesson plans to illustrate plan customization. Shown below is the instructional plan for a student who is deficient in electronics and has a low aptitude. It consists of 5 lessons and 51 steps. Some of the parameters of the procedures are shown in brackets after the name of the procedure.

Instructional Plan 1
(ELECTRONICS-DEFICIENT, LOW-APTITUDE student)

Lesson 1

1. *Orientation*
2. *Course-Overview*
3. *Structure-Overview* **[structure lower hoist]**
4. *Explain-Device-Simulation*
5. *Cycle-Overview* **[device subcycles lower hoist]**
6. *Motivate-Topic* **[normal behavior lower hoist]**
7. *Motivate-Topic* **[structure lower hoist]**
8. *Part-Identifications-by-Label* **[parts lower hoist]**
9. *Identify-Part-From-Role* **[parts lower hoist]**
10. *Part-Roles* **[parts lower hoist]**
11. *Multiple-Choice-Quiz* **[part roles lower hoist]**
12. *Summarize-Topic* **[structure lower hoist]**
13. *Short-Answer* **[structure lower hoist]**
14. *Motivate-Topic* **[operation lower hoist]**
15. *Cycle-Intro* **[device subcycles lower hoist]**
16. *Multiple-Choice-Quiz* **[device subcycles lower hoist]**
17. *Wrap-Up* **[lesson 1]**

Lesson 2

18. *Overview* **[lesson 2]**
19. *Explain-Part-Type-Operation* **[solenoid assemblies]**

20. *Short-Answer* [operation solenoid assemblies]
21. *Explain-Subcycles* [device subcycles lower hoist]
22. *Wrap-Up* [lesson 2]

Lesson 3

23. *Overview* [lesson 3]
24. *Predict-Part-State-Changes* [device subcycles lower hoist]
25. *Summarize-Topic* [operation lower hoist]
26. *Multiple-Choice-Quiz* [operation lower hoist]
27. *Summarize-Topic* [normal behavior lower hoist]
28. *Multiple-Choice-Quiz* [normal behavior lower hoist]
29. *Wrap-Up* [Lesson 3]

Lesson 4

30. *Overview* [lesson 4]
31. *Motivate-Topic* [predicting faulted behavior lower hoist]
32. *Explain-Fault-Types* [possible faults lower hoist]
33. *Short-Answer* [possible faults lower hoist]
34. *Explain-Predicting-Symptoms-from-Faults*
35. *Multiple-Choice-Quiz* [propagating faults to symptoms lower hoist]
36. *Summarize-Topic* [predicting faulted behavior lower hoist]
37. *Multiple-Choice-Quiz* [predicting faulted behavior lower hoist]
38. *Motivate-Topic* [abduction lower hoist]
39. *Generating-Plausible-Fault-Hypotheses* [generating the fault hypothesis set lower hoist]
40. *Short-Answer* [generating the fault hypothesis set lower hoist]
41. *Summarize-Topic* [abduction lower hoist]
42. *True-False-Quiz* [abduction lower hoist]
43. *Motivate-Topic* [device troubleshooting lower hoist]

44. *Explain-Troubleshooting-Strategy* **[troubleshooting strategy weighted split half troubleshooting]**
45. *Short-Answer* **[troubleshooting strategy weighted split half troubleshooting]**
46. *Summarize-Topic* **[device troubleshooting lower hoist]**
47. *Demo-Troubleshooting* **[troubleshooting strategy weighted split half troubleshooting]**
48. *Wrap-Up* **[lesson 4]**

Lesson 5

49. *Overview* **[lesson 5]**
50. *Monitor-Student-Troubleshooting* **[case-difficulty 3, wrong-tries-before-hint 2, prompt-menu yes, number-of-cases 5]**
51. *Course-Wrap-Up*

The next instructional plan that follows is for a student with the expected background in hydraulics, electronics, and mechanics and a high aptitude. It consists of only 3 lessons and only 26 steps:

Instructional Plan 2
(HAS-PREREQUISITES, HIGH-APTITUDE student)

Lesson 1

1. *Orientation*
2. *Course-Overview*
3. *Motivate-Topic* **[normal behavior lower hoist]**
4. *Explain-Device-Simulation*
5. *Part-Roles* **[parts lower hoist]**
6. *Multiple-Choice-Quiz* **[part roles lower hoist]**
7. *Motivate-Topic* **[operation lower hoist]**
8. *Cycle-Intro* **[device subcycles lower hoist]**
9. *Short-Answer* **[device subcycles lower hoist]**

10. *Explain-Subcycles* **[device subcycles lower hoist]**
11. *Wrap-Up* **lesson 1**

Lesson 2

12. *Overview* **[lesson 2]**
13. *Predict-Part-State-Changes* **[device subcycles lower hoist]**
14. *Summarize-Topic* **[operation lower hoist]**
15. *Multiple-Choice-Quiz* **[operation lower hoist]**
16. *Summariz· Topic* **[normal behavior lower hoist]**
17. *Short-Answer* **[normal behavior lower hoist]**
18. *Wrap-Up* **[lesson 2]**

Lesson 3

19. *Overview* **[lesson 3]**
20. *Motivate-Topic* **[device troubleshooting lower hoist]**
21. *Explain-Troubleshooting-Strategy* **[troubleshooting strategy weighted split half troubleshooting]**
22. *Multiple-Choice-Quiz* **[troubleshooting strategy weighted split half troubleshooting]**
23. *Summarize-Topic* **[troubleshooting strategy weighted split half troubleshooting]**
24. *Demo-Troubleshooting* **[troubleshooting strategy weighted split half troubleshooting]**
25. *Monitor-Student-Troubleshooting* **[case-difficulty 7, wrong-tries-before-hint 4, prompt-menu no, number-of-cases 2]**
26. Course-Wrap-Up

An explanation of the lesson plans and their differences is given below.

The steps (i.e., procedures) in the first lesson of the first instructional plan are:

1. *Orientation* - puts up a greeting for the student.
2. *Course-Overview* - provides an overview of what will be covered in

the lesson.

3. *Structure-Overview* - provides an overview of the structure of the lower hoist.

4. *Explain-Device-Simulation* - explains conventions used in the STEAMER color graphics display of the lower hoist assembly. (E.g., red color is used to depict high pressure and yellow color to depict low pressure.)

5. *Cycle-Overview* - provides an overview of the six subcycles of the lower hoist. For each subcycle the major parts affected are highlighted. Changes to just these parts are pointed out before and after each subcycle. Many details about changes to other parts are not mentioned. See Figure A-1 for color illustrations showing some of the steps in this procedure.

6. *Motivate-Topic* - explains to the student why it is important to understand how the lower hoist operates normally in order to perform effective troubleshooting.

7. *Motivate-Topic* - explains to the student why it is important to understand the structure of the lower hoist in order to perform effective troubleshooting. Note that this is the same procedure as the previous step, but with different parameters.

8. *Part-Identifications-by-Label* - highlights and describes each part of the lower hoist assembly.[9]

9. *Identify-Part-From-Role* - asks the student to use the mouse to point to each lower hoist part after the tutor names it and describes its function. When a part is not identified correctly the mistake is pointed out and the correct part is highlighted. The tutor continues to ask about all parts that were originally misidentified until the student has identified each part correctly.

10. *Part-Roles* - describes the role of each part in the operation of the lower hoist. (*Part-Identifications-by-Label* just describes what each part is and how it affects those parts immediately connected.)

11. *Multiple-Choice-Quiz* - asks several questions about the role of the parts of the lower hoist, to check if the previous step was effective.

12. *Summarize-Topic* - indicates that the tutor has finished discussing

---

[9]Actually, in these scenarios a subset of 8 parts - just those involved in just the first subcycle of operation - are used. The full set of about 40 parts can be used but then lessons are longer and the lesson plan generation process is slower, too.

the structure of the lower hoist.

13. *Short-Answer* - asks a few questions about the structure of the lower hoist to check the student's understanding.

14. *Motivate-Topic* - explains to the student why it is important to understand the operation of the lower hoist in order to perform effective troubleshooting. It also indicates a topic shift.

15. *Cycle-Intro* - provides a brief textual overview of lower hoist operation. It does not use the device simulation as *Cycle-Overview* does.

16. *Multiple-Choice-Quiz* - quizzes the student over the subcycles of the lower hoist.

17. *Wrap-Up* - summarizes what was covered in this first lesson and announces that the lesson is finished.

C. Blackboards and Knowledge Sources of BB-IP

This appendix describes the blackboards and knowledge sources of BB-IP-2. The blackboards are discussed first, then the knowledge sources.

C.1. The Blackboards

There are thirteen blackboards used in the BB1 implementation of the Blackboard Instructional Planner and the Lower Hoist Tutor. The four most relevant to the planner are:

1. *Instructional Plan* - the tutor's goals, intended activities, and intended procedures to carry out those activities.
2. *Planner Control* - the control phase, time remaining, problems noted during plan execution, diagnoses of plan problems, and plan edits made.
3. *Knowledge Sources* - plan refinement operators, the plan executor (*Execute Procedure* KS), monitoring demons, plan diagnosis methods, and plan repair methods.
4. *History* - Executed activities and procedures, student questions and requests, and assessments that have affected the student model.

The next three blackboards provide knowledge required for the student model, domain expert, and courseware modules (shown in Figure 1-3) of the Lower Hoist Tutor:

5. *Student Model* - Inferred cognitive stereotypes, inferred student aptitude, and an overlay student model consisting of certainty factors that annotate a (generated) graph of domain-specific skills.
6. *Device* - Parts breakdown, subcycles of operation, and part state changes in each subcycle.
7. *Curriculum* - Generic troubleshooting skills, questions for topics, and troubleshooting cases for student practice or tutor demonstration.

Some blackboards are typically present in any BB1 application. These generic BB1 blackboards [Garvey 87] are:

8. *Control Data* - The current agenda, history of events and KSARs executed.
9. *Control Plan* - Scheduling heuristics. The scheduler decides how to rate KSARs based on the contents of this blackboard.
10. *Concept* - General object taxonomy of BB1 objects.

The remaining blackboards are:

11. *PLAN Language* - A language framework that can be used to

express knowledge source actions of the Blackboard Instructional Planner. It provides a useful categorization of kinds of KSAR actions used in BB-IP. To enhance planner efficiency it is not currently used to express KSAR actions or for control.

12. *Tutoring Domain* - A taxonomy of objects in the tutoring domain (e.g., troubleshooting-case or student-question) tied into BB1's generic *Concept* knowledge base.

13. *Simple-Device* - A subset of the *Device* knowledge base. Has only 8 lower hoist parts instead of 42. Used for testing and demonstration purposes.

C.2. The Knowledge Sources

This section describes the 43 knowledge sources of the Blackboard Instructional Planner. They are listed in the order in which they are invoked, assuming top-down planning and complete plan elaboration. This order approximately corresponds to the control phases shown in Figure 5-3, if they are read left to right from the top down, and then counterclockwise around the circle. The plan generation KSes are:

1. *Initialize Planner* - determines what the top-level instructional objective is, how much time is available per lesson, whether lesson deadlines are preferences or constraints, and whether incremental planning is desired. When running a demonstration reasonable defaults are chosen otherwise these questions are asked via menu. Also computes the set of domain-specific skills by forming a cartesian product from the generic skills knowledge base and the device knowledge base.

2. *Meta-Level* - changes the control phase when no actions (KSARs) remain to be performed in the current control phase. Changing the control phase allows a different set of knowledge sources to trigger.

3. *Refine Objectives* - determines subordinate instructional objectives implied by the top-level instructional objective and adds them. Essentially, the subtree of domain-specific skills headed by the top-level instructional objective is copied.

4. *Assess Objectives* - determines to what degree the instructional objectives are currently satisfied based on the student model. It initializes the student model by giving a pre-instruction questionnaire.

5. *Propose Topics* - proposes topics to cover in order to achieve the pending instructional objectives. Each *topic* implicitly represents a COVER-TOPIC *activity* for the instructional plan.

6. *Prioritize Topics* - adds attributes to topics within the lesson plan indicating their priority.

7. *Propose Assessment Activities* - proposes activities to test the student's knowledge of the skills being taught.

8. *Propose Didactic Activities* - proposes practice or exploratory activities.

9. *Prioritize Activities* - adds attributes to activities (other than COVER-TOPIC activities) indicating their priority.

10. *Filter Activities* - selects the topics to cover and activities to perform to achieve the pending instructional objectives. Activities below a threshold are considered inappropriate or not important enough to retain.

11. *Sequence Activities* - imposes a total ordering on those activities selected for inclusion in the lesson plan. The ordering is determined by traversing the graph of instructional objectives, taking into account prerequisite links and partial order links between subtrees in the generic skills knowledge base of Figure 3-4.

12. *Refine Discourse* - critiques and improves the sequence of activities for the instructional plan. Introductions, overviews, and wrap-ups are added. MOTIVATE-TOPIC activities are moved to immediately precede the topic they motivate where necessary.

13. *Propose Actions* - selects candidate procedures for each activity. Each such procedure has a set of preconditions that must be satisfied for it to be a candidate.

14. *Filter Actions* - selects one procedure for each activity from the candidates found.

15. *Sequence Actions* - sequences actions to mirror the sequencing of their parent activities.

16. *Refine Actions* - critiques and improves the instructional plan now that procedures have been selected.

17. *Partition Lessons* - decides where to place lesson partitions so the instructional plan can be interpreted as a sequence of lesson plans.

The remaining KSes are used to carry out, monitor, and revise the initial lesson plan. The first is the workhorse of the Blackboard Instructional Planner:

18. *Execute Procedure* - executes the next procedure step of the current procedure, or starts the next procedure if the previous is finished.

The next set are monitoring demons that can interrupt execution and record some

complaint about the performance of the instructional plan. They monitor time and instructional objectives:

19. *Monitor Time* - monitors how well the current plan is adhering to its time constraints. It triggers if too little or too much time remains in the current lesson plan.

20. *Monitor Activities* - monitors student performance between procedure steps. It triggers if the student is not doing well enough on the current task over several procedure steps.

21. *Monitor Objectives* - triggers when assessment given *after* a presentation activity indicates the student did not learn as expected. *Monitor Objectives* is similar to *Monitor Activities*, but triggers only *after* an activity (of any kind) is completed.

22. *Monitor Prior Objectives* - triggers when assessment indicates an instructional objective that was previously believed to be satisfied now no longer appears satisfied.

23. *Monitor Future Objectives* - triggers when assessment indicates an instructional objective that was believed unsatisfied now appears satisfied, and that objective has pending procedures associated with it.

Once a complaint has been noted about the instructional plan the planner switches to the *Diagnose* control phase shown in Figure 5-3. The knowledge sources that can trigger in this control phase either make an assumption about the cause of the plan failure or actively try to diagnose the problem. A diagnosis must be made before plan repair is attempted. Here are the diagnosis KSes:

24. *Assume Lesson Too Long* - assumes that the remaining procedures cannot be finished in the time remaining.

25. *Assume Lesson Too Short* - similar, but assumes that too much time has been allotted for the remaining procedures.

26. *Assume Objectives Not Maintained* - assumes that prior instructional objectives have not been maintained, but should have been. The particular objectives not maintained are recorded as part of the diagnosis.

27. *Assume Objectives Satisfied* - assumes that pending instructional objectives have already been satisfied. It records which ones no longer need to be achieved.

28. *Assume Wrong Action Taken* - assumes that the reason why an instructional objective failed was because the wrong procedure was selected to achieve it. Can only trigger when there was another

choice that was passed over.

29. *Assume Wrong Parameters* - assumes that the reason the student is performing poorly on the current task, given by some procedure that lets a student practice a skill, is because the parameters for that procedure are set incorrectly. I.e., the tasks are too hard or the help given is insufficient.

30. *Test for Missing Prerequisite* - asks the student questions about a prerequisite skill for the current failed instructional objective. If the student does not answer most of the questions correctly then a partial diagnosis of plan failure is recorded. The diagnosis claims that the problem with the current instructional plan is that the student forgot or never properly learned that prerequisite skill.

31. *Finish Diagnosis* - records when the diagnosis appears sufficient to move on to the plan repair stage.

32. *Ask Student* - asks the student which of the prerequisite skills of the current failed instructional objective he thinks he needs help on.

Once the reason for the plan failure has been diagnosed then corrective action can be taken. These knowledge sources adjust parameters, add or delete lesson plan steps, or repartition the remaining instructional plan. These are the repair KSes:

33. *Repair Activities* - adjusts the parameters of the current procedure to be more appropriate to the student's perceived capabilities.

34. *Repair Actions* - chooses an alternate procedure to achieve a failed instructional objective.

35. *Reachieve Objective* - creates a new instructional plan fragment for a previously achieved instructional objective that has not been maintained. This plan fragment is elaborated using the plan generation knowledge sources and then spliced into the main lesson plan by the *Merge Plan Fragments* knowledge source.

36. *Repair Prerequisites* - creates new instructional plan fragments, one for each prerequisite objective that needs to be reviewed. These are elaborated using the plan generation knowledge sources and then spliced into the main lesson plan by the *Merge Plan Fragments* knowledge source. This KS is similar to *Reachieve Objective*, but also handles prerequisite material not covered in the lessons since the tutor believed the student knew these based on the pre-instruction questionnaire.

37. *Omit Redundant Procedures* - marks as deleted any procedures associated with pending instructional objectives that are now

believed to be already achieved.

38. *Replan Time* - removes the lesson partitions in the remainder of the instructional plan, then recomputes where they should be placed, and places them accordingly. This repartitioning effectively shuffles lesson partitions so that activities can be moved from the current lesson to the next if there is insufficient time in the current lesson.

The plan repairs initiated by the knowledge sources above may cause one or more plan patches to be created. *Plan patches* are modifications to the existing instructional p' .n made by adding, deleting, or modifying plan steps. When plan patches consist of steps to add to the plan there must be some means of integrating these smaller instructional plans into the main plan. The means of integration is provided by two knowledge sources:

39. *Merge Plan Fragments* - splices review plans into the main instructional plan, so that they will be executed next once the plan is resumed. The remainder of the unexecuted instructional plan follows the last inserted plan patch.

40. *Refine Plan Edits* - critiques the resulting plan, adding transitions where context shifts occur and removing redundant parameters in adjacent similar procedures.

Questions and requests are handled in a uniform manner since a question is simply a request to provide some information. These student interruptions are handled by two knowledge sources:

41. *Respond Now* - which creates plan patches to execute the procedures that accommodate the student's request(s). These plan patches are integrated into the main lesson plan by the *Merge Plan Fragments* and *Refine Plan Edits* knowledge sources.

42. *Defer* - defers the request until later. This knowledge source can only trigger if a procedure that would satisfy the student's request occurs later in the instructional plan.

The last knowledge source is:

43. *Focus Planner* - which limits the application of the previous knowledge sources. It is used in incremental planning to restrict the plan generation knowledge sources to only act on the next step of the lesson plan. It could be used to allow other parts of the plan to be developed opportunistically but has not been so used.

D. The PLAN Language Framework

The PLAN language framework, shown in Figure D-1, can be used to specify KS actions and control preferences. A control preference could be expressed using any PLAN verb such as GENERATE-LESSON-PLAN, which would match any of the KS actions that are terminal nodes under the GENERATE-LESSON-PLAN subtree. For example, PARTITION-LESSONS or PROPOSE-ACTIONS would match but MONITOR-TIME would not.

Rather than step through a sequence of steps expressing these control preferences[10], control phases are used to control the triggering of KSs. These control phases are shown in Figure 5-3 and described in Section 5. The planner operates in the same manner either way but the use of control phases is more efficient since it restricts agenda size and does not require the interpretation of language framework sentences. Some possibilities for opportunism are also sacrificed, but this loss appears negligible since BB-IP does not exploit opportunistic planning anyway.

The chief difference between the PLAN language framework and the earlier TUTOR language framework (shown in Figure F-2) is the lack of didactic actions in the newer framework. Instead of having didactic actions, such as MOTIVATE or EXPLAIN, the PLAN language framework has only planning actions such as REFINE-OBJECTIVES or EXECUTE-PROCEDURE. Because instructional plans have a separate representation from BB1 control plans, the ACT subtree of the TUTOR language is no longer necessary. Instead, the planning actions under the PLAN subtree of the TUTOR language framework have now evolved into a much richer set of planning actions that make up the new PLAN language framework. Since KSARs no longer represent all possible instructional actions (both planning and purely didactic) the size of the agenda is considerably reduced in BB-IP-2 and the new planner is much faster than BB-IP-1.

---

[10]These preferences are called *foci* in the jargon of BB1's control plans [Garvey 87].

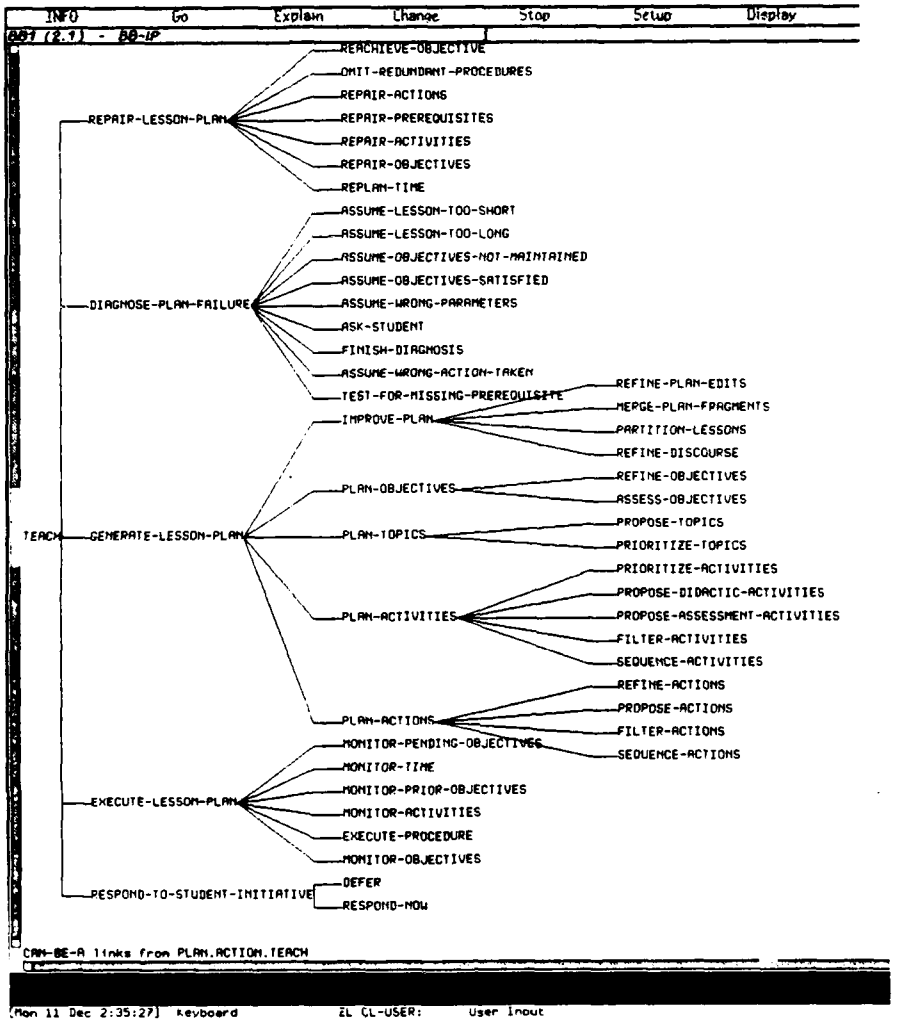


Figure D-1: The PLAN language framework

E. Comparison of SIIP and BB-IP Research

This section compares the SIIP planner architecture and planners implemented in it to the second version of the Blackboard Instructional Planner (BB-IP). The SIIP research is research on ITS and planner architectures by MacMillan and Sleeman [MacMillan 87]. The SIIP research encompasses

- *An ITS architecture* - called the BLACKBOARD-instructor.
- *A planner architecture* - called the SIIP architecture.
- *A planner* - implemented by MacMillan in the SIIP architecture. This planner will be referred to here as the SIIP planner.

The term SIIP can be confusing since it stands for "Self-Improving Instructional Planner" even though it refers to an architecture. Furthermore, although the BLACKBOARD-instructor *architecture* provides for a learning component, no *planner* was ever implemented in either the SIIP or BLACKBOARD-instructor architecture with such a self-improving capability.

In contrast to the SIIP research on architectures, the research presented in this report describes a *planner* (BB-IP) implemented in the BB1 version 2.1 blackboard architecture. Unlike the SIIP research, no claims are made that BB-IP is a generic planner architecture for all tutors, or that the prototype tutor that it has been embedded in (the Lower Hoist Tutor) is a generic ITS architecture. But to allow comparison of the SIIP research with the BB-IP research we will use the term *BB-IP architecture* to refer to the blackboards, levels, and semantics of blackboard objects that BB-IP uses in its BB1 implementation.

The SIIP architecture is basically the BB1 architecture (version 1) with the following additions:

- *Predefined levels* - for the instructional plan and control plan.
- *Predefined knowledge sources* - that can be used optionally or not in building instructional planners.
- *Database facilities* - allowing objects to be retrieved by specifying constraints they must satisfy.

The articles describing the SIIP research [MacMillan 87] and [MacMillan 88] can be confusing at times since they imply that "SIIP" is not only an architecture, but also a fully implemented planner with the capabilities implied by its name:

> In response to these and other deficiencies of instructional planners a generic system *architecture* based on the blackboard model was implemented. This self-improving instructional *planner* (SIIP) dynamically creates instructional plans, requests execution of these plans, replans, and improves its planning

behavior based on a student's responses to tutoring. ([MacMillan 87], page 17. Italics added here.)

In fact no such sophisticated planner was ever built. What was built was the modified blackboard architecture described below, along with sufficient knowledge sources to run a simple plan generation scenario (the SIIP planner).

There are four predefined levels on SIIP's instructional plan blackboard:

1. *Iobjective* - instructional objectives.
2. *Istrategy* - pedagogical strategies.
3. *Iprocedure* - instructional procedures.
4. *Iaction* - instructional actions.

The "I" stands for "instructional" in each level name. SIIP's control blackboard is essentially the control blackboard of BB1. That blackboard has levels to represent the current problem and BB1's control plan. This control plan is a sequence of steps where each step biases the scheduler to favor certain kinds of knowledge sources. For instance, a control plan could first favor knowledge sources that act on the Iobjective level, then the Istrategy level, then the Iprocedure level, and finally the Iaction level. This control plan would result in top-down plan expansion.

The overall ITS architecture in which SIIP was to run was called the BLACKBOARD-instructor. It is shown in Figure E-1 (this is Fig. 1 of [MacMillan 87]) and has the following components:

- *The planner* - the SIIP architecture and any planner implemented in it.
- *Student assessor* - a program that infers updates to the student model.
- *Expert model* - the domain expert.
- *Plan improvement experimenter* - a learning element to improve planner performance so planning improves for future students. (Never implemented.)
- *Instructional manager* - a program that executes actions from the planner and reports back their success or failure.

Each component runs in its own process and can communicate with other processes only via messages. This arrangement results in a complicated multiprocessing architecture where all of the following can happen simultaneously:

- The planner refines the instructional plan.
- The student assessor modifies the student model.

- The plan improvement experimenter changes the planner's knowledge base.
- The instructional manager executes an instructional action.

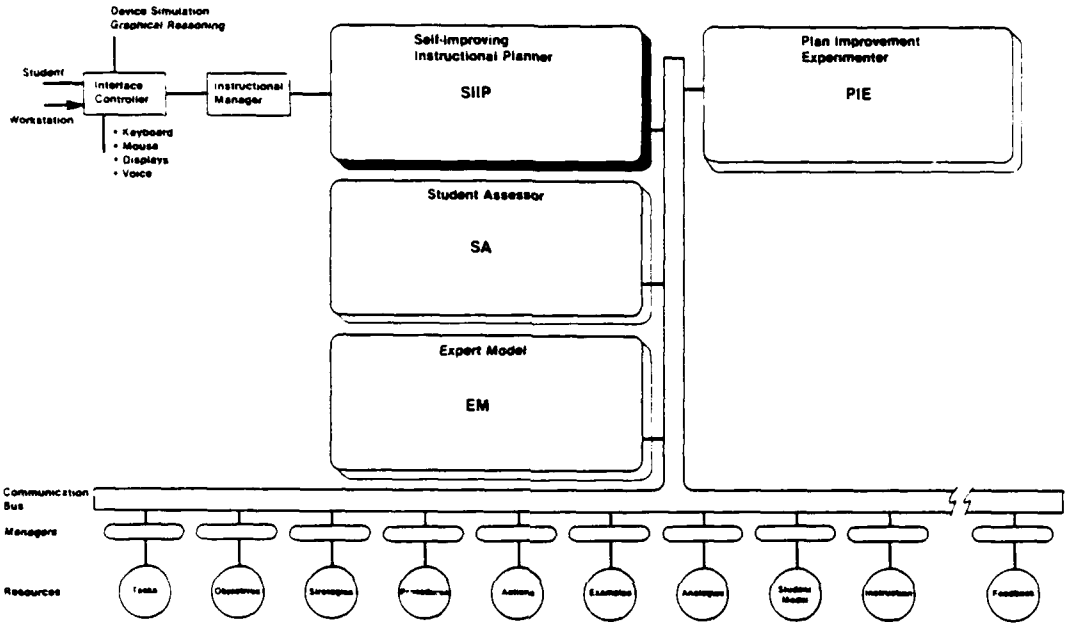


Figure E-1: The BLACKBOARD-instructor ITS architecture

Now we consider similarities and differences between the SIIP research and the BB-IP research. First, we consider the instructional plan representations and control plan representations of the SIIP architecture and the BB-IP architecture. Next, we consider how planners built in each architecture are intended to be integrated with other ITS components. Finally, we consider what was actually implemented in the SIIP research and the BB-IP research.

Comparing the planner architectures, the top levels of the instructional plan representations are similar but the other levels differ both in semantics and their implications for planner performance. We consider SIIP's levels and their BB-IP counterparts, if any:

- *Iobjective* - This SIIP level corresponds directly to the instructional objective level of BB-IP.
- *Istrategy* - SIIP's Istrategy level has no counterpart in BB-IP. Such a level appears unnecessary. It is ambiguous whether "strategy" in

"Istrategy" refers to tutorial modes of instruction (e.g., expository, exploratory, collaborative, case-method, etc.) or to high-level plans, which is the use of "strategy" in BB1 control plans. If the former is intended it is not clear what is gained by explicitly reasoning about the mode of instruction; if the latter is intended it appears to be an unnecessary layer of plan abstraction.

- *Iprocedure* - corresponds to BB-IP's *Activities* level since nodes posted on this level in SIIP only provide "an abstract specification" [MacMillan 87] of a procedure and do not represent actions that are directly executable.

- *Iaction* - does not correspond to any BB-IP level.

BB-IP omits this last level since it is not cost-effective to use a blackboard architecture to implement procedures that are largely algorithmic. No control reasoning is required so the overhead should be avoided. In fact, planners implemented in the SIIP planner architecture must *add* control reasoning to replicate simple programming structures such as conditional tests and loops.

SIIP uses the control blackboard of BB1 for its control reasoning while BB-IP provides an additional blackboard called *Planner Control* for this purpose. This latter approach is more efficient and perspicuous, but reduces the ability of BB-IP to plan opportunistically. It is more efficient since special purpose generic knowledge sources [Garvey 87] are not needed to sequence through BB1 control plans, and since BB-IP's use of control phases limits the number of knowledge sources triggered at any time. It is more perspicuous since the *Planner Control* blackboard can be used for making decisions on plan generation, execution, monitoring, and replanning while the BB1 control blackboard can be used to monitor the performance of the planner itself. For example, it could detect that a problem in the plan was not fixed by an earlier patch so a different approach to replanning should be used the second time the problem occurs. The BB1 control blackboard could also be used to determine how much time to devote to diagnosing a problem in the plan.

BB-IP does not provide an ITS shell similar to SIIP's BLACKBOARD-instructor. But since BB-IP is embedded in the Lower Hoist Tutor, as shown in Figure 1-3, we can compare how each planner would be embedded in a complete ITS. The BLACKBOARD-instructor provides the opportunity for parallelism among its modules at a considerable cost in computational and conceptual complexity. Not only must each module be implemented but an interface language must be defined - since they communicate only via messages - and synchronization must be ensured. Debugging can be difficult because of timing errors such as race conditions. BB-IP takes a much simpler approach. There is only one process - the planner - and all modules communicate via shared

blackboards. Thus they are more tightly interleaved and it would be cumbersome to separate them into different interacting processes.

Not all of the ITS components of the BLACKBOARD-instructor ITS shell were ever implemented. The shell, which was implemented, provides a process for each ITS component module but the programmer must supply the code to run within the processes. The *plan-improvement experimenter* module was never implemented in this sense for any planner built in the SIIP planner architecture. This module was intended to provide the learning element for the planner that would make it "self-improving" [MacMillan 87].

Now we consider the planners actually implemented. The following have been implemented in SIIP:

- *SIIP planner* - this is the default planner that produces the abstract scenario described in Section 4 of [MacMillan 87]. No domain is specified; basically a graph structure is grown on the plan blackboard and then abstract actions executed. No dynamic replanning occurs.
- *Instructional scenarios for the 25mm gun* - this is a demonstration of a tutor that would teach troubleshooting of common faults of the 25mm gun of the Bradley Fighting Vehicle. Again a simple graph structure is generated. The instructional plan is quite simple since most control decisions are made by the plan's actions. These actions are instructional procedures for presenting the troubleshooting cases and monitoring student performance.
- *Geometric concept demonstration tutor* - this is a demonstration tutor to teach geometric concepts such as rhomboid and trapezoid by showing examples and nonexamples of the concepts. Mitchell's version space algorithm [Mitchell 83] was used to represent the tutor's uncertainty of the student's acquired concept, and also to drive the selection of examples.

The first planner was implemented by MacMillan; the latter two by the author using the SIIP architecture.

All of these planners fail to generate customized lesson plans, address time management, handle student questions, or replan to remedy ineffective instruction. The current version of BB-IP handles all of these issues as described in this report. Thus, the BB-IP research addresses issues of plan customization, adaptive replanning, and mixed-initiative instruction in a planner-based tutor much more thoroughly than the SIIP research. In fact, the SIIP research does not specifically address the issue of handling unexpected student interruptions at all. Instead most of the discussion in [MacMillan 87] and [MacMillan 88] focusses on abstract planning issues - skeletal planning, constraint-based planning,

hierarchical planning, and opportunistic planning - without demonstrating precisely how these can be advantageously used to realize plan customization, adaptive replanning, or mixed-initiative instruction.

F. Comparison of BB-IP-1 and BB-IP-2

The immediate precursor to the current version of BB-IP, and immediate successor to the SIIP research, is the first version of BB-IP. The first will be called BB-IP-1 and the second BB-IP-2 to avoid confusion. A detailed comparison between the two is presented to show how deficiencies in BB-IP-1 led to BB-IP-2.

BB-IP-1 is directly patterned on PROTEAN [Hayes-Roth 87b], a blackboard application built using BB1 Version 2. PROTEAN determines the three-dimensional structure of protein molecules based on constraints that include nuclear magnetic resonance measurements. It uses a language called ACCORD [Hayes-Roth, et al 87] to represent different possible problem-solving actions. An example action in ACCORD might be:

YOKE HELIX-76 TO HELIX-19 WITH CONSTRAINT-SET-22

which would apply the set of constraints called CONSTRAINT-SET-22 to HELIX-76 and HELIX-19 to restrict their possible locations in three dimensional space. With this language control plans can be built that control the application of problem-solving operators (the KSARs). For example, a control plan might be:

1. ANCHOR MOST-CONSTRAINING ALPHA-HELIX
2. ORIENT OBJECT TO ANCHOR
3. YOKE OBJECT-1 TO OBJECT-2 WITH CONSTRAINT-SET

This would first pick an object to be considered the origin of the molecule. That object, called an *anchor*, must be the alpha-helix whose position is most constrained. The second step in the control plan determines the possible positions of all other objects relative to this anchor. The third step applies constraints between pairs of these objects to further restrict their positions. The problem-solving action

YOKE HELIX-76 TO HELIX-19 WITH CONSTRAINT-SET-22

would match the third step. Partial matching is defined by a taxonomy of verbs and objects. The taxonomy of verbs for the ACCORD language is shown in Figure F-1.

Analogously, BB-IP-1 is built over a problem-solving language for instructional planning problems. This language is called TUTOR and is shown in Figure F-2. Now BB1 control plans are used to express instructional plans rather than assembly plans. No separate plan representation or execution mechanism is required since BB1 provides these. Furthermore, BB1's explanation mechanisms can be used to explain why one action (either instructional or planning) is performed rather than another. An example of a simple instructional plan written

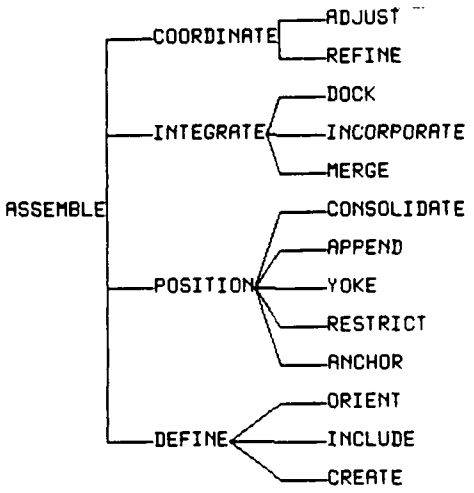


Figure F-1: The ACCORD language framework

in TUTOR would be:

1. SELECT LESSON-OBJECTIVE
2. MOTIVATE TOPIC
3. OVERVIEW TOPIC
4. EXPLAIN TOPIC
5. GIVE-TRUE-FALSE-TEST FOR TOPIC

Complete details of BB-IP-1 and TUTOR are presented in [Murray 89a].

BB-IP-1 was the first planner developed in the blackboard architecture in this research whose domain was a realistic training application and which addressed issues of plan generation, mixed initiative instruction, and adaptive planning. But there was room for considerable improvement. It suffered from slow operation due to:

- *Control reasoning overhead* - Many blackboard cycles were required for the execution of each instructional action. Additional knowledge sources, called *generic control knowledge sources* [Garvey 87], must sequence through the steps in BB1 control plans and determine when each step finishes. These generic control knowledge sources perform no instructional actions and increase the size of the agenda.
- *Agenda size* - Many KSARs could be on the agenda at any time. All possible instructional actions were considered along with different possible instantiations.

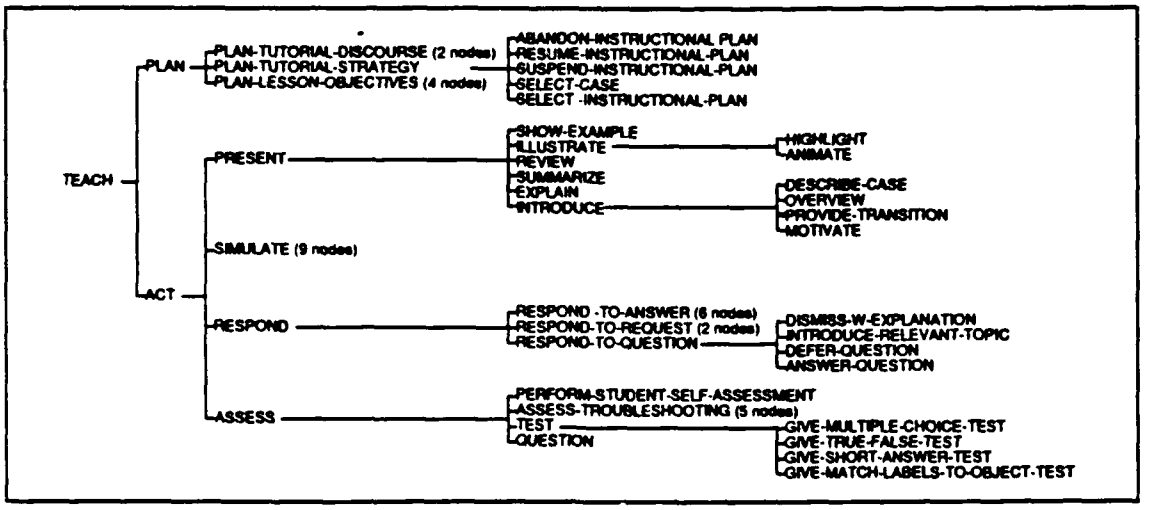


Figure F-2: The TUTOR language framework

- *Language framework overhead* - The problem-solving language TUTOR is large (i.e., has many verbs, objects, and modifiers). The TUTOR action sentence of a KS must be partially matched to the current control plan step when that KS is triggered. This matching is required to rate the KSARs generated so the scheduler can decide which one is the most appropriate to execute next.

Other problems with the first planner were its:

- *Overly simplistic plans* - such as
  1. Present topic
  2. Assess topic
  3. Respond
- *Lack of a realistic testbed* - Although the target application is a real one, the planner itself is disembodied, and its instructional actions were only simulated.
- *Overly fine granularity* - Some KSAR actions were very fine grained and could be immediately executed (e.g., highlighting an icon). Others were less fine grained and represented actions that might take several minutes to perform (e.g., give a multiple choice test or explain a topic). Because of the overhead of the blackboard system it was not worthwhile to reason about control for low-level actions. The overhead is only justified for reasoning about higher-level actions (e.g., how to repair an ineffective instructional plan) where knowledge can be fruitfully applied. Lower-level actions such as presenting test questions do not require the sophisticated application

of knowledge. Instead, they should be handled by straightforward procedural control mechanisms.

BB-IP-2 addressed all these concerns. It operates near real-time. The 51-step instructional plan shown in Appendix B is generated in about 95 seconds. For most procedures, there is only a small pause of a second or two between procedure steps. The blackboard cycle time is typically one to two seconds for BB-IP-2 compared to eight to fourteen seconds for BB-IP-1. The improved performance is due to

- *Decreased control reasoning overhead* - The number of control knowledge sources has been reduced since control plans no longer represent instructional plans.
- *Smaller agenda size* - Typically, 5 or less KSARs are executable at any time. BB-IP-1 would commonly have as many as 15 KSARs executable.
- *Language framework overhead* - No language framework is used.
- *Improved BB1 implementation* - CONSing and the use of graphics during execution has been reduced.

Other improvements in BB-IP-2 include:

- *More sophisticated plan representation* - The plan representation is much richer: it represents plan rationale, lesson partitioning, and expected time for activities. Plan steps (instructional procedures) can be sophisticated procedures that may take several minutes to execute while still being interruptible.
- *More Realistic testbed* - The second planner is embedded in the prototype Lower Hoist Tutor. A STEAMER-based student interface is coupled with the planner. All instructional actions are implemented. As the instructional plan is executed tutorial text is displayed and the STEAMER interface is used for part highlighting, text display, student input, and selective animation of device parts.
- *More Appropriate Action granularity* - Use of procedures as plan steps eliminates the need for control reasoning for simple procedural control.

Other key differences in the design of the second planner are:

- *KSes correspond to lesson planning skills.* No KSes correspond to domain instructional actions such as MOTIVATE-TOPIC.
- *Emphasis on lesson planning.* Control knowledge is now applied at a higher level - at the level of lesson planning and replanning. Although both planners perform some discourse and some lesson

planning, BB-IP-1 focuses on discourse planning while BB-IP-2 focuses on lesson planning.

- *Plan generation rather than plan selection* - BB-IP-1 is basically a skeletal planner that selects discourse plans from a library. The second version generates lesson plans and uses pre-stored procedures as higher-level actions to carry out lower level discourse actions.
- *Plan patching* - BB-IP-2 can patch plans; the first planner did not.
- *Separate representation of control plans for the instructional planner and control plans for the blackboard system* - The lesson plan being constructed and the control plan that controls how it is being constructed are both represented as BB1 domain blackboards in the second version.
- *Curriculum planning is implemented* - BB-IP-2 generates a sequence of lesson plans whereas BB-IP-1 assumes only a single lesson.
- *Time management is addressed* - BB-IP-2 estimates the time required for each of its plan actions (i.e., instructional procedure executions) and uses these estimates to determine lesson partitions. It also replans when time runs out. BB-IP-1 assumes an indefinite lesson length and does not monitor time.

Both planners share the following differences with the SIIP research:

- *Emphasis on the planner not the architecture* - the BB-IP-1 and BB-IP-2 research focussed on exploring how best to implement a dynamic instructional planner in an existing architecture rather than implementing a new architecture.
- *Shared class of instructional problems* - both BB-IP-1 and BB-IP-2 are intended for the same class of problems: troubleshooting complex hydraulic-electronic-mechanical devices, where a mental model is required for effective troubleshooting and a STEAMER device simulation is available for this purpose. SIIP is not tailored for any specific class of instructional problem.
- *Lack of multiprocessing* - both BB-IP planners eliminate the multiprocessing used in the BLACKBOARD-instructor. Instructional planning is difficult enough without having to plan and act at the same time the student model is updated.
- *Lack of a learning element* - since the planning problem alone is difficult enough and should be addressed first. Both BB-IP planners were adaptive since they improved their performance as the student model became more refined, but neither had a learning element for improving planner performance from one student to another.

References

[Barr, et al 76] A. Barr, M. Beard, and R. Atkinson.
The Computer as a Tutorial Laboratory.
International Journal of Man-Machine Studies (8):567 - 596, 1976.

[Brecht 89] B. J. Brecht, G. I. McCalla, J. E. Greer.
Planning the Content of Instruction.
In *Proceedings of the 4th International Conference on AI and Education*, pages 32 - 41. IOS, Springfield, VA., May, 1989.

[Breuker 87] J. Breuker, R. Winkels, and J. Sandberg.
A Shell for Intelligent Help Systems.
In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 167 - 173. August, 1987.

[Brown 75] J. S. Brown, R. R. Burton, and A. G. Bell.
SOPHIE: a Step towards a Reactive Learning Environment.
International Journal of Man-Machine Studies 7:675 - 696, 1975.

[Burton and Brown 79]
R. R. Burton, and J. S. Brown.
An Investigation of Computer Coaching for Informal Learning Activities.
International Journal of Man-Machine Studies (11):5 - 24, 1979.

[Carbonell 70] J. R. Carbonell.
Mixed-Initiative Man-Computer Instructional Dialogues.
PhD thesis, Massachusetts Institute of Technology, 1970.

[Carr and Goldstein 77]
B. Carr, and I. P. Goldstein.
Overlays: a Theory of Modeling for Computer-aided Instruction.
Technical Report AI Lab Memo 406, Massachusetts Institute of Technology, 1977.

[Chandrasekaran 89]
B. Chandrasekaran.
Editor's Note.
Blackboard Architectures and Applications.
Academic Press, Inc., Boston, MA, 1989, pages v - vi.

[Clancey 79] W. Clancey.
Tutoring Rules for guiding a case method dialogue.
International Journal of Man-Machine Studies (11):25 - 49, 1979.

[Clancey 87] W. Clancey.
Knowledge-based Tutoring.
The MIT Press, 1987.

[Corkill 87] Daniel D. Corkill, Kevin Q. Gallagher, and Philip M. Johnson.
Achieving Flexibility, Efficiency, and Generality in Blackboard Architectures.
In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 18 - 23. University of Massachusetts, Morgan Kaufmann Publishers, Inc., Los Altos, CA., July, 1987.

[Derry 88] S. J. Derry, L. Hawkes, U. Zeigler.
A Plan-based Opportunistic Architecture for Intelligent Tutoring.
In *Proceedings, ITS-88.* Montreal, Canada, June, 1988.

[Erman, et al 80] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy.
The Hearsay-II Speech-understanding System: Integrating Knowledge to Resolve Uncertainty.
Computing Surveys (12):213 - 253, 1980.

[Feigenbaum 88] E. Feigenbaum.
Foreword.
Blackboard Systems.
Addison-Wesley Publishing Company, Menlo Park, CA, 1988, pages v - viii.

[Fikes and Nilsson 71]
R. E. Fikes, and N. J. Nilsson.
STRIPS: a New Approach to the Application of Theorem Proving to Problem Solving.
Artificial Intelligence 2:189-208, 1971.

[Garvey 87] A. Garvey, M. Hewett, M. V. Johnson Jr., R. Schulman and B. Hayes-Roth.
BB1 User Manual - Common LISP Version 2.0.
Technical Report KSL-86-61, Stanford Knowledge Systems Laboratory, August, 1987.

[Hayes-Roth 84] B. Hayes-Roth.
BB1: An Architecture for Blackboard Systems that Control, Explain, and Learn about their own Behavior.
Technical Report HPP 84-16, Knowledge Systems Laboratory, Stanford University, 1984.

[Hayes-Roth 85] B. Hayes-Roth.
A Blackboard Architecture for Control.
Artificial Intelligence 26(3):251 - 321, 1985.

[Hayes-Roth 87a] B. Hayes-Roth.
Blackboard Systems.
Encyclopedia of Artificial Intelligence.
Wiley-Interscience Publication, New York, 1987, pages 73 - 80.

[Hayes-Roth 87b] B. Hayes-Roth, B. Buchanan, O. Lichtarge, M. Hewett, R. Altman, J. Brinkley, C. Cornelius, B. Duncan, and O. Jardetzky.
PROTEAN: Deriving Protein Structure from Constraints.
Blackboard Systems.
Addison-Wesley, 1987.

[Hayes-Roth, et al 87]
B. Hayes-Roth, A. Garvey, M. V. Johnson, and M. Hewett.
A Modular and Layered Environment for Reasoning about Action.
Technical Report KSL 86-38, Stanford University, April, 1987.

[Hollan 84] J. D. Hollan, E. L. Hutchins, and L. Weitzman.
STEAMER: an interactive inspectable simulation-based training system.
AI Magazine 5(2):15 - 27, 1984.

[Johnson 85] W. L. Johnson.
Intention-Based Diagnosis of Errors in Novice Programs.
PhD thesis, Yale, May, 1985.

[Joyce 86] B. R. Joyce and M. Weil.
Models of Teaching.
Prentice Hall, Englewood Cliffs, NJ., 1986.

[Lesgold 88] A. Lesgold, S. Lajoie, M. Bunzo, and G. Eggan.
SHERLOCK: A Coached Practice Environment for an Electronics Troubleshooting Job.
Technical Report, Learning Research and Development Center, University of Pittsburgh, Pittsburgh, Pennsylvania, March, 1988.

[MacMillan 87] S. A. Macmillan, and D. H. Sleeman.
An Architecture for a Self-improving Instructional Planner for Intelligent Tutoring Systems.
Computational Intelligence 3(1):17 - 27, 1987.

[MacMillan 88] S. A. Macmillan, D. Emme, M. Berkowitz.
Instructional Planners: Lessons Learned.
Intelligent Tutoring Systems: Lessons Learned.
Lawrence Erlbaum Associates, Inc., 1988.

[Mitchell 83] T. M. Mitchell, P. E. Utgoff, R. Banerji.
Learning by Experimentation: Acquiring and Refining Problem-solving Heuristics.
Machine Learning.
Tioga Publishing Company, 1983.

[Moore 89] J. D. Moore and W. R. Swartout.
A Reactive Approach to Explanation.
In *Eleventh International Joint Conference on Artificial Intelligence*, pages 1504 - 1510. Morgan Kaufmann Publishers, Inc., Los Altos, CA., August, 1989.

[Murray 88a] W. R. Murray.
Control for Intelligent Tutoring Systems: A Comparison of Blackboard Architectures and Discourse Management Networks.
Technical Report R-6267, FMC Corporation, September, 1988.
To appear in *Machine-Mediated Learning* 3(1), 1990.

[Murray 88b] W. R. Murray.
Automatic Program Debugging for Intelligent Tutoring Systems.
Pitman Publishing, London, 1988.

[Murray 88c] W. R. Murray.
Dynamic Instructional Planning in the BB1 Blackboard Architecture.
Technical Report R-6168, FMC Corporation, August, 1988.

[Murray 89a] W. R. Murray.
Dynamic Instructional Planning in the BB1 Blackboard Architecture.
Blackboard Architectures and Applications.
Academic Press, Inc., 1989, pages 455 - 480.

[Murray 89b] W. R. Murray.
Control for Intelligent Tutoring Systems: a Blackboard-based Dynamic Instructional Planner.
In *Proceedings of the 4th International Conference on AI and Education*, pages 150 - 168. IOS, Springfield, VA., May, 1989.
Reprinted in *AI Communications*, Vol. 2, No. 2, June 1989.

[Murray 89c] W. R. Murray.
Control for Intelligent Tutoring Systems: a Blackboard-based Dynamic Instructional Planner.
AI Communications 2(2):41 - 57, June, 1989.
First appeared in Proceedings of the 4th International Conference on AI and Education, May, 1989.

[Nirenburg 89] S. Nirenburg, V. Lesser, and E. Nyberg.
A Reactive Approach to Explanation.
In *Eleventh International Joint Conference on Artificial Intelligence*, pages 1524 - 1530. Morgan Kaufmann Publishers, Inc., Los Altos, CA., August, 1989.

[Papert 80] S. Papert.
Mindstorms: Children, Computers, and Powerful Ideas.
Basic Books, New York, 1980.

[Peachey 86] D. R. Peachey, and G. I. McCalla.
Using Planning Techniques in Intelligent Tutoring Systems.
International Journal of Man-Machine Studies 24:77 - 98, 1986.

[Reimann 89] P. Reimann.
Modeling Scientific Discovery Learning Processes with Adaptive Production Systems.
In *Proceedings of the 4th International Conference on AI and Education*, pages 218 - 227. IOS, Springfield, VA., May, 1989.

[Reiser 85] B. Reiser, J. Anderson, and R. Farrell.
Dynamic Student Modelling in an Intelligent Tutor for Lisp Programming.
In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 8 - 14. 1985.

[Rich 79] E. Rich.
User Modeling via Stereotypes.
Cognitive Science 3:355 - 366, 1979.

[Russell 85] S. Russell.
The Compleat Guide to MRS.
Technical Report KSL-85-12, Stanford Knowledge Systems Laboratory, June, 1985.

[Russell 88a] D. M. Russell.
IDE: The Interpreter.
Intelligent Tutoring Systems: Lessons Learned.
Lawrence Erlbaum Associates, Inc., 1988.

[Russell 88b] D. M. Russell, T. P. Moran, and D. S. Jordan.
The Instructional-Design Environment.
Intelligent Tutoring Systems: Lessons Learned.
Lawrence Erlbaum Associates, Inc., 1988.

[Sacerdoti 77] E. D. Sacerdoti.
A Structure for Plans and Behavior.
Elsevier North-Holland, 1977.

[Shute 86] V. Shute and R. Glaser.
An Intelligent Tutoring System for Exploring Principles of Economics.
Technical Report, Learning Research and Development Center, University of Pittsburgh, Pittsburgh, Pennsylvania, 1986.

[Stansfield 76] J. C. Stansfield, B. Carr, and I. P. Goldstein.
Wumpus Advisor I: a First Implementation of a Program that Tutors Logical and Probabilistic Reasoning Skills.
Technical Report AI Lab Memo 381, Massachusetts Institute of Technology, 1976.

[Stevens and Collins 77]
A. L Stevens, and A. Collins.
The Goal Structure of a Socratic Tutor.
In *Proceedings of the National ACM Conference*, pages 256 - 263. Association for Computing Machinery, 1977.

[Towne 89] Douglas M. Towne, Allen Munro, Quentin A. Pizzini, David S. Surmon, and James Wogulis.
ONR Final Report: Intelligent Maintenance Training Technology.
Technical Report Technical Report No. 113, Behavioral Technology Laboratories, University of Southern California, September, 1989.

[Wilkins 84] D. E. Wilkins.
Domain-independent Planning: Representation and Plan Generation.
Artificial Intelligence 22(3):269 - 301, 1984.

[Wilkins 85] D. E. Wilkins.
Recovering from Execution Errors in SIPE.
Computational Intelligence 1:33 - 45, 1985.

[Wilkins 88] D. E. Wilkins.
Practical Planning.
Morgan Kaufmann Publishers, Inc., San Mateo, CA., 1988.

[Woolf 87] B. P. Woolf.
Representing Complex Knowledge in an Intelligent Machine Tutor.
Computational Intelligence 3(1):45 - 55, 1987.

[Woolf, et al 84] B. P. Woolf, and D. D. McDonald.
Building a Computer Tutor: Design Issues.
IEEE Computer 17(9):61 - 73, 1984.

Personnel Analysis Division,
AF/MPXA
5C360, The Pentagon
Washington, DC 20330

Air Force Human Resources Lab
AFHRL/MPD
Brooks, AFB, TX 78235

AFOSR,
Life Sciences Directorate
Bolling Air Force Base
Washington, DC 20332

Technical Director, ARI
5001 Eisenhower Avenue
Alexandria, VA 22333

Technical Director,
Army Human Engineering Lab
ATTN: SLCHE-D
Aberdeen Proving Ground
MD 21005-5001

Dr. Beth Adelson
Department of Computer Science
Tufts University
Medford, MA 02155

Dr. Robert Ahlers
Code N711
Human Factors Laboratory
Naval Training Systems Center
Orlando, FL 32813

Technical Director
Air Force Human Resources Lab.
Brooks AFB, TX 78236-5601

Dr. John R. Anderson
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Stephen J. Andriole, Chairman
Department of Information Systems
and Systems Engineering
George Mason University
4400 University Drive
Fairfax, VA 22030

Dr. Patricia Baggett
School of Education
610 E. University, Rm 1302D
University of Michigan
Ann Arbor, MI 48109-1259

Dr. Eva L. Baker
UCLA Center for the Study
of Evaluation
145 Moore Hall
University of California
Los Angeles, CA 90024

Dr. Meryl S. Baker
Navy Personnel R&D Center
San Diego, CA 92152-6800

Prof. Dott. Bruno G. Bara
Unita di ricerca di
Intelligenza Artificiale
Universita di Milano
20122 Milano - via F. Sforza 23
ITALY

Dr. Gautam Biswas
Department of Computer Science
Box 1688, Station B
Vanderbilt University
Nashville, TN 37235

Dr. John Black
Teachers College, Box 8
Columbia University
525 West 120th Street
New York, NY 10027

Dr. Deborah A. Boehm-Davis
Department of Psychology
George Mason University
4400 University Drive
Fairfax, VA 22030

Dr. Jeff Bonar
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. David Bowers
Rensis Likert Associates
3001 S. State Street
Ann Arbor, MI 48104-7352

Dr. Robert Breaux
Code 7B
Naval Training Systems Center
Orlando, FL 32813-7100

Dr. John S. Brown
XEROX Palo Alto Research
Center
3333 Coyote Road
Palo Alto, CA 94304

Dr. John T. Bruer
James S. McDonnell Foundation
1034 So. Brentwood Blvd., Ste. 1610
St. Louis, MO 63117

Dr. Bruce Buchanan
Computer Science Department
Stanford University
Stanford, CA 94305

Lt. Col. Hugh Burns
AFHRL/IDI
Brooks AFB, TX 78235

Assistant for Long Range Rqmts.
CNO Executive Panel (Op-OOK)
4401 Ford Avenue
Alexandria, VA 22302-0268

Dr. Joanne Capper, Director
Center for Research into Practice
1718 Connecticut Ave., N.W.
Washington, DC 20009

Dr. Pat Carpenter
Carnegie-Mellon University
Department of Psychology
Pittsburgh, PA 15213

Dr. John M. Carroll
IBM Watson Research Center
User Interface Institute
P.O. Box 704
Yorktown Heights, NY 10598

Director, Manpower Program
Center for Naval Analyses
4401 Ford Avenue
P.O. Box 16268
Alexandria, VA 22302-0268

Center for Personnel
Security Research
Suite E, Bldg. 455
99 Pacific Street
Monterey, CA 93940-2481

Professor Chu Tien-Chen
Mathematics Department
National Taiwan University
Taipei, TAIWAN

Dr. Michelene Chi
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Raymond E. Christal
UES LAMP Science Advisor
AFHRL/MOEL
Brooks AFB, TX 78235

Dr. William Clancey
Institute for Research
on Learning
3333 Coyote Hill Road
Palo Alto, CA 94304

Dr. Allan M. Collins
Bolt Beranek & Newman, Inc.
10 Moulton Street
Cambridge, MA 02238

Dr. Stanley Collyer
Office of Naval Technology
Code 222
800 N. Quincy Street
Arlington, VA 22217-5000

Dr. Albert T. Corbett
Department of Psychology
Carnegie-Mellon University
Pittsburgh, PA 15213

CAPT P. Michael Curran
Chief of Naval Operations
OP-939
Pentagon
Washington, DC 20350-2000

Brian Dallman
Training Technology Branch
3400 TCHTW/TTGXC
Lowry AFB, CO 80230-5000

Dr. Robert B. Davis
Curriculum Laboratory
(Education)
University of Illinois
Urbana, IL 61801

Chief, Survey & Market
Analysis Division
Defense Manpower Data Center
1600 Wilson Blvd., #400
Arlington, VA 22209-2593

Defense Technical Info. Ctr.
Attn: TC
Cameron Station, Bldg. 5
Alexandria, VA 22314
(12 copies)

Deputy Dir. Military Personnel
Policy Division
Office of the DCNO (MPT)
(Op-13B)
Department of the Navy
Washington, DC 20370-2000

Deputy Director Total Force
Training & Education Division
Office of the DCNO (MPT)
(Op-11B)
Department of the Navy
Washington, DC 20370-2000

Head, Leadership Branch
Naval Military Personnel Command
Attn: LCDR E. Marits, NMPC-621
Department of the Navy
Washington, DC 20370-5620

Head, Military Compensation
Policy Branch
Office of the DCNO (MPT)
(Op-134)
Department of the Navy
Washington, DC 20370-2000

Dr. Andrea di Sessa
University of California
School of Education
Tolman Hall
Berkeley, CA 94720

ERIC Facility-Acquisitions
4350 East-West Hwy, Suite 1100
Bethesda, MD 20814-4475

Dr. Martha Evans
Dept. of Computer Science
Illinois Institute of Technology
Chicago, IL 60616

Dr. Marshall J. Farr, Consultant
Cognitive & Instructional Sciences
2520 North Vernon Street
Arlington, VA 22207

Dr. Paul Feltovich
Southern Illinois Univ.
School of Medicine
P.O. Box 3926
Springfield, IL 62708

Dr. Gerhard Fischer
University of Colorado
Department of Computer Science
Boulder, CO 80309

Dr. Kenneth D. Forbus
University of Illinois
Department of Computer Science
1304 West Springfield Avenue
Urbana, IL 61801

Dr. Barbara A. Fox
University of Colorado
Department of Linguistics
Boulder, CO 80309

Dr. John R. Frederiksen
BBN Laboratories
10 Moulton Street
Cambridge, MA 02238

Dr. Michael Friendly
Psychology Department
York University
Toronto ONT
CAŃADA M3J 1P3

Julie A. Gadsden
Information Technology
Applications Division
Admiralty Research Establishment
Portsdown, Portsmouth PO6 4AA
UNITED KINGDOM

Eric Gaussens
Research & Development Dept.
Framentec S.A.
Tour Fiat Cedex 16
Paris la Defense
F. 92084
France

Dr. Dedre Gentner
University of Illinois
Department of Psychology
603 E. Daniel St.
Champaign, IL 61820

Dr. Meg Gerrard
Psychology Department
Iowa State University
Ames, IA 50010

Dr. Robert Glaser
Learning Research &
Development Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Sam Glucksberg
Department of Psychology
Princeton University
Princeton, NJ 08540

Prof. Clark Glymour
Dept. of Philosophy
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. Sherrie Gott
AFHRL/MOMJ
Brooks AFB, TX 78235-5601

Dr. T. Govindaraj
Georgia Institute of
Technology
School of Industrial
and Systems Engineering
Atlanta, GA 30332-0205

Dr. Jordan Grafman
Neuropsychology Section
Medical Neurology
Branch-NINCDS
Building 10, Room 5C416
Bethesda, MD 20892

Dr. James G. Greeno
School of Education
Stanford University
Room 311
Stanford, CA 94305

Dr. Henry M. Halff
Halff Resources, Inc.
4918 33rd Road, North
Arlington, VA 22207

Dr. Bruce W. Hamill
Research Center
The Johns Hopkins University
Applied Physics Laboratory
Johns Hopkins Road
Laurel, MD 20707

Dr. Barbara Hayes-Roth
Knowledge Systems Laboratory
Stanford University
701 Welch Road
Palo Alto, CA 94304

Dr. James Hollan
NPRDC, UCSD
Code 501
San Diego, CA 92152

Dr. Keith Holyoak
Department of Psychology
University of California
Los Angeles, CA 90024

Ms. Julia S. Hough
Lawrence Erlbaum Associates
110 W. Harvey Street
Philadelphia, PA 19144

Dr. Ed Hutchins
Intelligent Systems Group
Institute for
Cognitive Science (C-015)
UCSD
La Jolla, CA 92093

Dr. Janet Jackson
Rijksuniversiteit Groningen
Biologisch Centrum, Vleugel D
Kerklaan 30, 9751 NN Haren
The NETHERLANDS

Dr. Claude Janvier
Universite' du Quebec a Montreal
P.O. Box 8888, succ: A"
Montreal, Quebec H3C 3P8
CANADA

Dr. Robin Jeffries
Hewlett-Packard Laboratories, 3L
P.O. Box 10490
Palo Alto, CA 94303-0971

Prof. David W. Johnson
Cooperative Learning Center
University of Minnesota
150 Pillsbury Dr., S.E.
Minneapolis, MN 55455

Dr. Marcel Just
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Daniel Kahneman
Department of Psychology
University of California
Berkeley, CA 94720

Dr. Milton S. Katz
European Science Coordination
Office
U.S. Army Research Institute
Box 65
FPO New York 09510-1500

Dr. Wendy Kellogg
IBM T. J. Watson Research Ctr.
P.O. Box 704
Yorktown Heights, NY 10598

Dr. Jeffery L. Kennington
School of Engineering &
Applied Sciences
Southern Methodist University
Dallas, TX 75275

Dr. David Kieras
Technical Communication Program
TIDAL Bldg., 2360 Bonisteel Blvd.
University of Michigan
Ann Arbor, MI 48109-2108

Dr. Janet L. Kolodner
Georgia Institute of Technology
School of Information &
Computer Science
Atlanta, GA 30332

Dr. Kenneth Kotovsky
Community College of
Allegheny County
808 Ridge Avenue
Pittsburgh, PA 15212

Dr. Benjamin Kuipers
University of Texas at Austin
Department of Computer Sciences
Taylor Hall 2.124
Austin, Texas 78712

Dr. Jill Larkin
Carnegie-Mellon University
Department of Psychology
Pittsburgh, PA 15213

Commander J.M. LaRocco
Naval School of Health Sciences
National Naval Medical Center
Bldg. 141
Washington, DC 20814-5033

Dr. Robert W. Lawler
Matthews 118
Purdue University
West Lafayette, IN 47907

Dr. Alan M. Lesgold
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Jim Levin
Department of
Educational Psychology
210 Education Building
1310 South Sixth Street
Champaign, IL 61820-6990

Dr. John Levine
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Clayton Lewis
University of Colorado
Department of Computer Science
Campus Box 430
Boulder, CO 80309

Science and Technology Division
Library of Congress
Washington, DC 20540

Vern M. Malec
NPRDC, Code 52
San Diego, CA 92152-6800

Dr. Jane Malin
Mail Code EF5
NASA Johnson Space Center
Houston, TX 77058

Dr. William L. Maloy
Naval Education and Training
Program Support Activity
Code 047
Building 2435
Pensacola, FL 32509-5000

Dr. Elaine Marsh
Naval Center for Applied Research
in Artificial Intelligence
Naval Research Laboratory
Code 5510
Washington, DC 20375-5000

Dr. Sandra P. Marshall
Dept. of Psychology
San Diego State University
San Diego, CA 92182

Dr. Manton M. Matthews
Department of Computer Science
University of South Carolina
Columbia, SC 29208

Dr. Joseph C. McLachlan
Code 52
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. James McMichael
Technical Director
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Barbara Means
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

Prof. George A. Miller
Dept. of Psychology
Princeton University
Princeton, NJ 08544

Dr. James R. Miller
MCC
3500 W. Balcones Center Dr.
Austin, TX 78759

Dr. William Montague
NPRDC Code 13
San Diego, CA 92152-6800

Dr. Judy Moracco
Code CEL-MP53
Washington Navy Yard
Bldg. 200
Washington, DC 20374

Dr. Randy Mumaw
Training Research Division
HumRRO
1100 S. Washington
Alexandria, VA 22314

Dr. Allen Munro
Behavioral Technology
Laboratories - USC
1845 S. Elena Ave., 4th Floor
Redondo Beach, CA 90277

Deputy Technical Director
NPRDC Code 01A
San Diego, CA 92152-6800

Director, Human Factors
& Organizational Systems Lab,
NPRDC (Code 07)
San Diego, CA 92152-6800

Director, Manpower and Personnel
Laboratory,
NPRDC (Code 06)
San Diego, CA 92152-6800

Director, Training Laboratory,
NPRDC (Code 05)
San Diego, CA 92152-6800

Head, Fleet Liaison Dept.
NPRDC (Code 31)
San Diego, CA 92152-6800

Head, Human Factors Dept.
NPRDC (Code 41)
San Diego, CA 92152-6800

Head, Manpower Systems Dept.
NPRDC (Code 61)
San Diego, CA 92152-6800

Head, Personnel Systems Dept.
NPRDC (Code 62)
San Diego, CA 92152-6800

Head, Testing Systems Dept.
NPRDC (Code 63)
San Diego, CA 92152-6800

Head, Training Systems Dept.
NPRDC (Code 52)
San Diego, CA 92152-6800

Head, Training Tech. Dept.
NPRDC (Code 51)
San Diego, CA 92152-6800

Library
NPRDC
Code P201L
San Diego, CA 92152-6800

Spec. Asst. for Research, Experimental & Academic Programs,
NTTC (Code 016)
NAS Memphis (75)
Millington, TN 38054

Assistant Chief of Staff
for Research, Development,
Test, and Evaluation
Naval Education and
Training Command (N-5)
NAS Pensacola, FL 32508

Director, Instructional Dvlpmt.
and Educational Pgm. Support Dept.
Naval Education & Training Pgm.
Management Support Activity
(NETPMSA)
Pensacola, FL 32509

Technical Director
Naval Health Research Center
P.O. Box 85122
San Diego, CA 92138-9174

Director, Recreational Svcs. Dept.
Naval Military Personnel Command
(N-651C)
1300 Wilson Blvd., Room 932
Arlington, VA 22209

Chairman, Department of
Administrative Sciences
Code 54
Naval Postgraduate School
Monterey, CA 93943-5100

Chairman, Department of
Operations Research
Code 55
Naval Postgraduate School
Monterey, CA 93943-5100

Commanding Officer,
Naval Research Laboratory
Code 2627
Washington, DC 20390

Deputy Director Manpower,
Personnel and Training Div.
Naval Sea Systems Command
Attn: Code CEL-MP63
Washington, DC 20362

Head, Human Factors Laboratory
Naval Training Systems Ctr.
Code 71
Orlando, FL 32813-7100

Library
Naval Training Systems Center
Orlando, FL 32813

Library
Naval War College
Newport, RI 02940

Commanding Officer
Navy Personnel R&D Ctr.
San Diego, CA 92152-6800

Technical Director
Navy Personnel R&D Center
San Diego, CA 92152-6800

Director, Research & Analysis Div.
Navy Recruiting Command
(Code 223)
4015 Wilson Blvd., Room 215
Arlington, VA 22203-1991

Dr. Donald A. Norman (C-015)
Institute for Cognitive Science
University of California
La Jolla, CA 92093

Special Assistant for Marine
Corps Matters,
ONR Code 00MC
800 N. Quincy St.
Arlington, VA 22217-5000

Chairman, MPT R&D Committee
Office of the Chief of
Naval Research
Code 222
Arlington, VA 22217-5000

Director, Biological/Human
Factors Division (Code 125)
Office of the Chief of
Naval Research
Arlington, VA 22217-5000

Director, Navy Family Support Pgm
Office of the DCNO (MPT) (Op-156)
1300 Wilson Blvd., Room 828
Arlington, VA 22209

Office of Naval Research,
Code 1142
800 N. Quincy St.
Arlington, VA 22217-5000

Office of Naval Research,
Code 1142BI
800 N. Quincy Street
Arlington, VA 22217-5000

Office of Naval Research,
Code 1142CS
800 N. Quincy Street
Arlington, VA 22217-5000
(6 Copies)

Office of Naval Research,
Code 1142PS
800 N. Quincy Street
Arlington, VA 22217-5000

Biological Intelligence
Code 1142BI
Office of Naval Research
Arlington, VA 22217-5000

Cognitive Science
Code 1142CS
Office of Naval Research
Arlington, VA 22217-5000

Director, Cognitive &
Neural Sciences (Code 1142)
Office of Naval Research
Arlington, VA 22217-5000

Director, Life Sciences
Code 114
Office of Naval Research
Arlington, VA 22217-5000

Director Research Programs
Office of Naval Research
Code 11
Arlington, VA 22217-5000

Mathematics
Code 1111MA
Office of Naval Research
Arlington, VA 22217-5000

Psychologist
Office of Naval Research
Branch Office, London
Box 39
FPO New York, NY 09510

Psychologist
Office of Naval Research
Detachment
1030 E. Green Street
Pasadena, CA 91106-2485

Dr. Harold F. O'Neil, Jr.
School of Education - WPH 801
Department of Educational
Psychology & Technology
University of Southern California
Los Angeles, CA 90089-0031

Dr. Judith Orasanu
Basic Research Office
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Military Assistant for Training and
Personnel Technology,
OUSD (R & E)
Room 3D129, The Pentagon
Washington, DC 20301-3080

Assistant for Manpower and Training
Office of the CNO (Op-987H)
5D772, The Pentagon
Washington, DC 20350

Head, Manpower, Personnel,
and Training Branch
Office of the CNO (Op-813)
4A478, The Pentagon
Washington, DC 20350-1000

Dr. Douglas Pearce
1133 Sheppard W
Box 2000
Downsview, Ontario
CANADA M3M 3B9

Office of the Deputy Assistant
Secretary of the Navy
Manpower & Reserve Affairs
5D800, The Pentagon
Washington, DC 20350-1000

Dr. Tjeerd Plomp
Twente University of Technology
Department of Education
P.O. Box 217
7500 AE ENSCHEDE
THE NETHERLANDS

Dr. Martha Polson
Department of Psychology
University of Colorado
Boulder, CO 80309-0345

Dr. Steven E. Poltrock
MCC
3500 West Balcones Center Dr.
Austin, TX 78759-6509

Dr. Harry E. Pople
University of Pittsburgh
Decision Systems Laboratory
1360 Scaife Hall
Pittsburgh, PA 15261

Dr. Joseph Psotka
ATTN: PERI-IC
Army Research Institute
5001 Eisenhower Ave.
Alexandria, VA 22333-5600

Mr. Paul S. Rau
Code U-33
Naval Surface Weapons Center
White Oak Laboratory
Silver Spring, MD 20903

Dr. Steve Reder
Northwest Regional
Educational Laboratory
400 Lindsay Bldg.
710 S.W. Second Ave.
Portland, OR 97204

Dr. James A. Reggia
University of Maryland
School of Medicine
Department of Neurology
22 South Greene Street
Baltimore, MD 21201

Dr. J. Wesley Regian
AFHRL/IDI
Brooks AFB, TX 78235

Dr. Fred Reif
Physics Department
University of California
Berkeley, CA 94720

Dr. Brian Reiser
Department of Psychology
Green Hall
Princeton University
Princeton, NJ 08540

Dr. Gilbert Ricard
Mail Stop K02-14
Grumman Aircraft Systems
Bethpage, NY 11787

Dr. J. Jeffrey Richardson
Center for Applied AI
College of Business
University of Colorado
Boulder, CO 80309-0419

Dr. Linda G. Roberts
Science, Education, and
Transportation Program
Office of Technology Assessment
Congress of the United States
Washington, DC 20510

Dr. William B. Rouse
Search Technology, Inc.
4725 Peachtree Corners Circle
Suite 200
Norcross, GA 30092

Dr. Eduardo Salas
Human Factors Division
Code 712
Naval Training Systems Ctr.
Orlando, FL 32813-7100

Dr. Walter Schneider
Learning R&D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Janet W. Schofield
816 LRDC Building
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Miriam Schustack
Code 52
Navy Personnel R & D Center
San Diego, CA 92152-6800

Dr. Judith W. Segal
OERI
555 New Jersey Ave., NW
Washington, DC 20208

Dr. Colleen M. Seifert
Institute for Cognitive Science
Mail Code C-015
University of California, San Diego
La Jolla, CA 92093

Dr. Ben Shneiderman
Dept. of Computer Science
University of Maryland
College Park, MD 20742

Dr. Randall Shumaker
Naval Research Laboratory
Code 5510
4555 Overlook Avenue, S.W.
Washington, DC 20375-5000

Dr. Edward E. Smith
Department of Psychology
University of Michigan
330 Packard Road
Ann Arbor, MI 48103

Program Director
Manpower Reearch &
Advisory Services
Smithsonian Institution
801 N. Pitt St., Suite 120
Alexandria, VA 22314-1713

Dr. Al Smode
Naval Training Systems Ctr.
Code 71
Orlando, FL 32813-7100

Dr. Elliot Soloway
Yale University
Computer Science Department
P.O. Box 2158
New Haven, CT 06520

Dr. Richard C. Sorensen
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Kathryn T. Spoehr
Brown University
Department of Psychology
Providence, RI 02912

Dr. Kurt Steuck
AFHRL/IDI
Brooks AFB
San Antonio, TX 78235-5601

Dr. Albert Stevens
Bolt Beranek & Newman, Inc.
10 Moulton St.
Cambridge, MA 02238

Dr. John Tangney
AFOSR/NL, Bldg. 410
Bolling AFB, DC 20332-6448

Dr. Kikumi Tatsuoka
CERL
252 Engineering Research
Laboratory
103 S. Mathews Avenue
Urbana, IL 61801

Dr. Perry W. Thorndyke
FMC Corporation
Central Engineering Labs
1205 Coleman Avenue, Box 580
Santa Clara, CA 95052

Dr. Martin A. Tolcott
3001 Veazey Terr., N.W.
Apt. 1617
Washington, DC 20008

Douglas M. Towne
University of Southern California
Behavioral Technology Laboratories
250 N. Harbor Drive, Suite #309
Redondo Beach, CA. 90277

Headquarters, U. S. Marine Corps
Code MPI-20
Washington, DC 20380

Dr. Jerry Vogt
Navy Personnel R&D Center
Code 51
San Diego, CA 92152-6800

Dr. Ralph Wachter
JHU-APL
Johns Hopkins Road
Laurel, MD 20707

Dr. Keith T. Wescourt
FMC Corporation
Central Engineering Labs
1205 Coleman Ave., Box 580
Santa Clara, CA 95052

Dr. Barbara White
BBN Laboratories
10 Moulton Street
Cambridge, MA 02238

Dr. Kent E. Williams
Inst. for Simulation and Training
University of Central Florida
P.O. Box 25000
Orlando, FL 32816-0544

Dr. Robert A. Wisher
U.S. Army Institute for the
Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Dr. Martin F. Wiskoff
Defense Manpower Data Center
550 Camino El Estero
Suite 200
Monterey, CA 93943-3231

Dr. Wallace Wulfeck, III
Navy Personnel R&D Center
Code 51
San Diego, CA 92152-6800

Dr. Masoud Yazdani
Dept. of Computer Science
University of Exeter
Prince of Wales Road
Exeter EX44PT
ENGLAND

Dr. Joseph L. Young
National Science Foundation
Room 320
1800 G Street, N.W.
Washington, DC 20550